ARMY RESEARCH LABORATORY

A Human Steering Model Used to Control Vehicle Dynamics Models

by Richard J. Pearson and Peter J. Fazio

ARL-TR-3102 December 2003

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-TR-3102 **December 2003**

A Human Steering Model Used to Control Vehicle Dynamics Models

Richard J. Pearson and Peter J. Fazio
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
December 2003	Final	October 2001 to September 2002

4. TITLE AND SUBTITLE
A Human Steering Model Used to Control Vehicle Dynamics Models

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Richard J. Pearson and Peter J. Fazio (both of ARL)

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
Weapons and Materials Research Directorate
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-3102

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report documents the modification of a legacy human steering model with the addition of a new three-dimensional driving sensor model. The three-dimensional sensor model was developed to replace the two-dimensional sensor model in the original code. The report also covers the integration of the resulting modified human steering model with a vehicle dynamics model developed in commercial off-the-shelf (COTS) simulation software. The COTS software is called the Dynamic Analysis and Design System (DADS). In this study, the legacy model was adapted to interface with the current version of DADS. The modified human steering model was interfaced with a model of the high mobility multipurpose wheeled vehicle (HMMWV) in DADS. Experimental data exist for driving tests that use the HMMWV. These data were used to validate the integrated human steering model.

15. SUBJECT TERMS
control; DADS; human steering model; vehicle dynamics

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Richard J. Pearson
Unclassified	Unclassified	Unclassified	UL	95	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6676

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

1.1 Legacy Human Steering Model

Human steering models have been used by the automotive industry as part of their vehicle dynamics modeling efforts. The U.S. Army Tank Automotive Command contracted with the University of Michigan Transportation Research Institute to modify one of the industrial models to work with military vehicles. The steering model was modified to work with wheeled and tracked military vehicles. We could model conventional front wheel steering, front and rear wheel steering, and skid steering by changing input parameters in the model. At the conclusion of the contract, a report entitled "Development of Driver/Vehicle Steering Interaction Models for Dynamic Analysis" (*1*) was written about the new human steering model. The existing or "legacy" human steering model that served as the basis for the current study was taken from reference (*1*).

1.2 Software Architecture

This study added a new sensor model to the steering model and linked the modified steering model to a vehicle dynamics simulation in the dynamics analysis and design system (DADS[1]). The original steering model contained a simple two-dimensional (2-D) sensor model that did not take into account the pitch and yaw of the vehicle or height of objects in the virtual environment. The new sensor model is three-dimensional (3-D) and completely coupled to the vehicle dynamics model. Figure 1 shows the basic process flow of the new integrated model. The external vehicle dynamics model was developed within DADS as a complex multi-body simulation. The desired path is input as a series of 3-D obstacles that border the route, similar to concrete "Jersey" barriers. The sensor model measures the distance from the vehicle to the surrounding obstacles. Based on the range map developed by the sensor, the steering model plans or previews a path for the vehicle. The steering model tries to maintain the vehicle equidistant from obstacles to the right and left. The steering model contains a simplified internal model of the vehicle's dynamics. The internal model is a linearized, 2-D model. The internal model represents the human driver's perception of how the vehicle behaves and is used to predict the vehicle's state at some time in the future. The steering model takes as input the current state of the vehicle, previewed path, and the predicted vehicle state during a preview interval. The steering model determines the difference between the previewed and the predicted path of the vehicle during the preview interval called the "path error". The steering model calculates a steering command that minimizes the "path error". The steering command is passed to the external vehicle model and produces control forces that change the vehicle's state.

[1]DADS™ is a trademark of LMS (not an acronym).

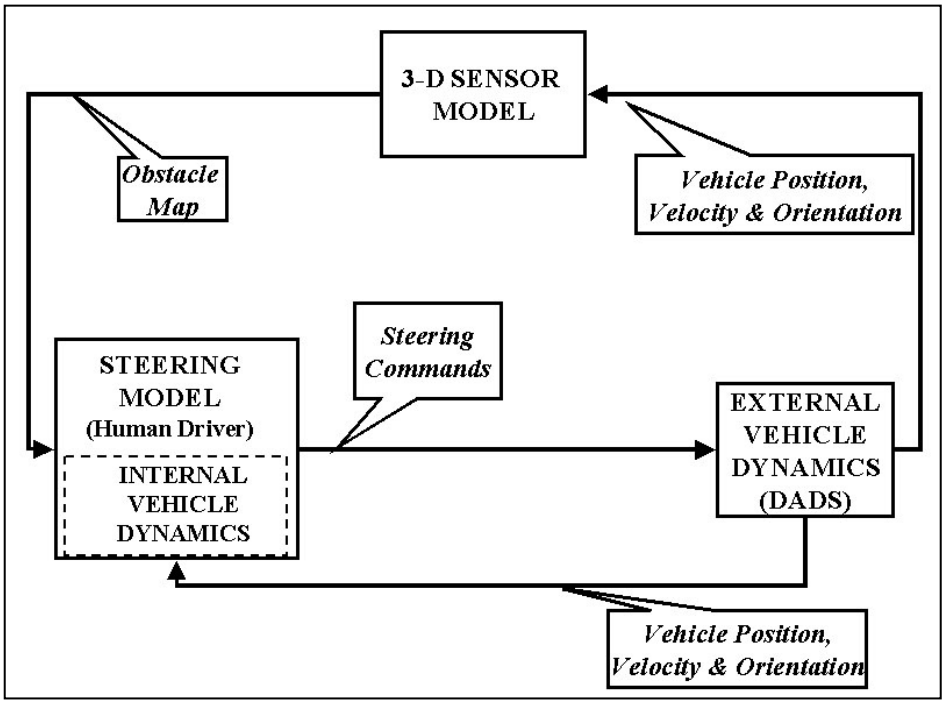

Figure 1. Steering model process flow.

A complete listing of the modified human steering model and its interface to DADS is given in appendix A. The control portion of the human steering model has not been significantly modified except for comments in the code. The comments have been modified to bring them in line with the derivation of the control algorithm given in this report. The subroutines taken directly from the reference (*1*) are Avoid_Obstacles, CALCRS, CALCTH, CHECKRTH, GMPRD, NEWDRIVER, and TRANS. The other subroutines are either new or are major revisions of subroutines presented in reference (*1*).

2. Human Steering Simulation

2.1 The Steering Control Loop

Figure 2 shows a block diagram of the control algorithm used in the steering subroutines "NEWDRIVER". The parameters used in the steering algorithm are initialized by "HUMAN_STEERING" which then calls "NEWDRIVER". The algorithm in "NEWDRIVER" calculates a steering control signal over a time interval called the preview time. The control signal is calculated to minimize the difference or error between the planned or previewed path and the vehicle's actual path. The steering control signal is delayed by a fix amount of time before it is sent to the external vehicle dynamics model. The fixed delay represents the human driver's neuromuscular delay in generating the command signal, i.e., turning the steering wheel.

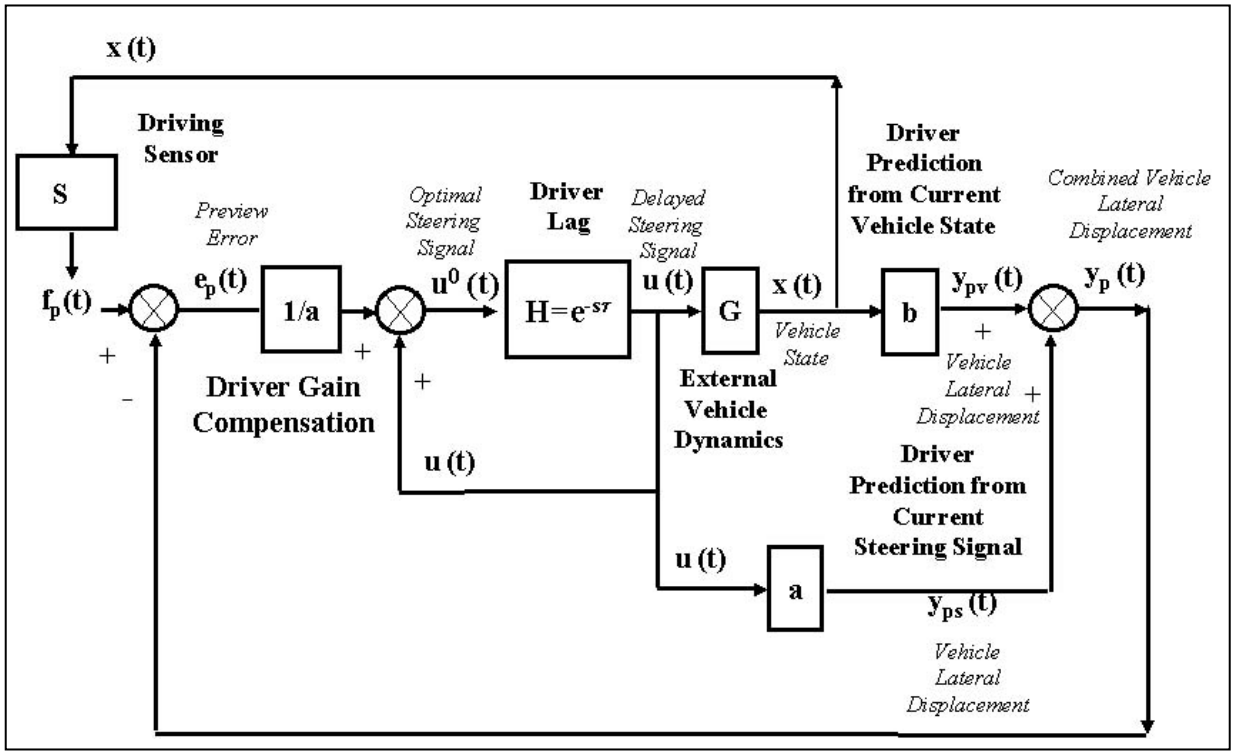

Figure 2. Control algorithm block diagram.

The course through the algorithm starts at the left with the previewed vehicle's path information received from the vehicle's sensors. The previewed path is subtracted from the path predicted by the driver. The difference is the error to be minimized by the control algorithm. The error is acted upon by the gain to produce a revised steering signal. The revision is summed with the current steering signal. The new steering signal is delayed by the neuromuscular delay to become the current steering signal. The current steering signal is sent to the external vehicle dynamics model where it produces control forces. The external vehicle dynamics model calculates the effects of the control forces on the vehicle state. The current steering signal is also sent backward to be summed with the steering signal update. The current steering signal is used by the driver to predict the vehicle path during the preview interval. The current vehicle state is used in a separate calculation by the driver to predict the vehicle's path during the preview interval. These two predictions of the path are summed to produce the current predicted path during the preview interval. The current predicted path is fed back and summed with the current previewed path. Finally, the vehicle state is sent to the sensor model where it is used to revise the current previewed path information.

2.2 Derivation of the Steering Algorithm

Reference (*1*) gives a derivation of the steering algorithm, but the listing of the code in appendix A shows a somewhat different algorithm. The algorithm that was actually used to calculate the

steering signal is derived in this report. In the derivation of the control algorithm, the system is assumed to be linear. The equation motion of the vehicle is

$$\begin{aligned} \dot{x} &= \boldsymbol{F}\, x + g\,\mathsf{u} \\ y &= m^{\mathsf{T}} x \end{aligned} \tag{1}$$

in which

x is the vehicle state vector,

$\dot{x}$ is the vector containing the time derivative of the vehicle state,

g is the control coefficient vector,

u is the scalar steering command, and

$\boldsymbol{F}$ is the 4 by 4 system matrix.

Note: Parameters in italics are *vectors*, parameters in bold and italics are ***matrices***, and parameters written in New Gothic MT font are scalars.

The vehicle's state vector x and the fundamental matrix $\mathbf{F}$ are given by the following:

$$x = \begin{bmatrix} \mathsf{Y} \\ \mathsf{v} \\ \mathsf{r} \\ \psi \end{bmatrix} \tag{2}$$

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 & 0 & \mathsf{U} \\ 0 & -2(C_{af} + C_{ar})/\mathsf{mU} & 2(\mathsf{b}C_{af} - \mathsf{a}C_{ar})/\mathsf{mU} - \mathsf{U} & 0 \\ 0 & 2(\mathsf{b}C_{af} - \mathsf{a}C_{ar})/\mathsf{IU} & 2(\mathsf{a}^2 C_{af} - \mathsf{b}^2 C_{ar})/\mathsf{IU} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3}$$

In equations 2 and 3, the parameters are defined as follows:

Y = vehicle's lateral displacement

v = vehicle's lateral velocity

ψ = vehicle's yaw angle

r = vehicle's yaw rate

U = vehicle's forward velocity (assumed to be constant)

m = vehicle's mass

I = vehicle's moment of inertia about the vertical axis

a = distance from the vehicle's center of gravity (cg) to the front axle centerline

b = distance from the vehicle's cg to the rear axle centerline

C_{af} = front tire cornering stiffness

C_{ar} = the rear tire cornering stiffness

f_f = the front wheel steering angle

f_r = the rear wheel steering angle

k = the proportion of f_f

The vehicle's dynamic transition vector *g* is given by the following:

$$g = \begin{bmatrix} 0 \\ A[2(C_{af} + kC_{af})/m] + B/m \\ C[2(C_{af} + kbC_{af})/I] + D/I \\ 0 \end{bmatrix} \tag{4}$$

In equation 4, the A, B, C, and D parameters are used to model different types of steering (*1*). When A and C are set to 1.0 and B, D, and k are set to 0.0, the standard front-wheel-only steering is modeled. If front and rear steering is modeled, A, B, C, and D remain the same but k assumes some value between 0 and 1. In reference (*1*), tests were conducted with k = 0.75. When A, B, and C are set to 0.0 and D is set to 1.0, skid steering is modeled. In the modeling of a tracked vehicle, C_{af} and C_{ar} are interpreted as an equivalent lateral force generated by track elements because of side slip.

The first part of equation 1, the equations of motion, can be put in the following form:

$$\dot{x} = \boldsymbol{A} \cdot x + h(t) \tag{5}$$

with the initial condition given by the following:

$$x(0) = x_0 \tag{6}$$

in which x_0 is some known initial position.

In equation 5,

$$\boldsymbol{A} = \boldsymbol{F}$$
$$h(t) = g \cdot u(t)$$

The fundamental matrix $\boldsymbol{\phi}$ is defined sothat (*2*)

$$\begin{aligned} \dot{\boldsymbol{\phi}} &= \boldsymbol{A} \cdot \boldsymbol{\phi} \\ \boldsymbol{\phi}(0) &= \boldsymbol{I} \end{aligned} \tag{7}$$

Laplace transforms will be used to solve equation 7 for $\boldsymbol{\phi}$. Taking the Laplace transform of each side of equation 7 yields the following (*3*):

$$\ell\left[\dot{\boldsymbol{\phi}}\right] = \ell\left[\mathbf{A} \cdot \boldsymbol{\phi}\right] \tag{8}$$

When the initial condition from equation 7 is used in equation 8, it can be rewritten as follows:

$$s\ell[\phi] - \phi(0) = s\ell[\phi] - \boldsymbol{I} = \boldsymbol{A} \cdot \ell[\phi] \tag{9}$$

in which *s* is some positive real number.

Solving equation 9 for ϕ and substituting for $\boldsymbol{A}$, we get the following:

$$\phi = \ell^{-1}\left[-(\boldsymbol{A} - s\boldsymbol{I})^{-1}\right] = \ell^{-1}\left[-(\boldsymbol{F} - s\boldsymbol{I})^{-1}\right] \tag{10}$$

In equation 10, ℓ^{-1} is the inverse Laplace transform.

Returning to equation 5 and taking the Laplace transform of both sides gives the following:

$$\begin{aligned} \ell[\dot{x}] &= \ell[\boldsymbol{A} \cdot x + h(t)] \\ s\,X(s) - x_0 &= \boldsymbol{A} \cdot x_0 + H(s) \end{aligned} \tag{11}$$

in which H is the Laplace transform of h

Solving equation 11 for $X(s)$ gives the following:

$$X(s) = \left[-(\boldsymbol{A} - s\boldsymbol{I})^{-1}\right] \cdot x_0 + \left[-(\boldsymbol{A} - s\boldsymbol{I})^{-1}\right] \cdot H(s) \tag{12}$$

Taking the inverse Laplace transform of each side of equation 12 and solving for *X* gives the following:

$$\begin{aligned} \ell^{-1}[\boldsymbol{X}(s)] &= \ell^{-1}\left\{\left[-(\boldsymbol{A} - s\boldsymbol{I})^{-1}\right] \cdot x_0 + \left[-(\boldsymbol{A} - s\boldsymbol{I})^{-1}\right] \cdot H(s)\right\} \\ x(t) &= \ell^{-1}\left\{\left[-(\boldsymbol{A} - s\boldsymbol{I})^{-1}\right] \cdot x_0\right\} + \ell^{-1}\left\{\left[-(\boldsymbol{A} - s\boldsymbol{I})^{-1}\right] \cdot H(s)\right\} \end{aligned} \tag{13}$$

With equation 10, the first left-hand term can be rewritten as follows:

$$\ell^{-1}\left\{-(\boldsymbol{A} - s\boldsymbol{I})^{-1}\right\} \cdot x_0 = \phi \cdot x_0 \tag{14}$$

With the convolution theorem and equation 10, the second term of equation 13 can be written as

$$\ell^{-1}\left\{\left[-(\boldsymbol{A} - s\boldsymbol{I})^{-1}\right] \cdot H\right\} = \int_0^t \phi(t - \xi) \cdot h(t)\,d\xi \tag{15}$$

Using equations 14 and 15 in equation 13 and using the definition of *h* gives

$$x(t) = \phi_0 \cdot x_0 + \int_0^t \phi(t - \xi) \cdot h(\xi)\,d\xi = \phi_0 \cdot x_0 + \int_0^t \phi(t - \xi) \cdot g \cdot u(\xi)\,d\xi \tag{16}$$

Solving for the vehicle's lateral displacement *y* gives the following:

$$y = m^T \cdot x(t) = m^T \cdot \left[\phi \cdot x_0 + \int_0^t \phi(t - \xi) \cdot g \cdot u(\xi)\,d\xi\right] \tag{17}$$

In equation 17, m^{T} is the constant observer vector

$$m^{\mathsf{T}} = [1 \quad 0 \quad 0 \quad 0]$$

The error *J*, between the previewed vehicle path $f(\eta)$ and the predicted vehicle path can be written as follows(*1*)

$$J = \frac{1}{(t_p - t_0)} \int_{t_0}^{t_p} \{f(\eta) - y(\eta)\}^2 \, d\eta \tag{18}$$

The minimum path error with respect to the steering control u satisfies

$$\frac{dJ}{du} = \frac{d}{du}\left\{ \frac{1}{(t_p - t_0)} \int_{t_0}^{t_p} [f(\eta) - y(\eta)]^2 \, d\eta \right\} = 0 \tag{19}$$

Using equation 17 to substitute for $y(\eta)$ in equation 19,

$$\frac{dJ}{du} = \frac{d}{du}\left\{ \frac{1}{(t_p - t_0)} \int_{t_0}^{t_p} \left[f(\eta) - m^{\mathsf{T}} \cdot \left(\phi \cdot x_0 + \int_0^t \phi(t-\xi) \cdot g \cdot u(\xi) d\xi \right) \right]^2 d\eta \right\} = 0 \tag{20}$$

Differentiating equation 20,

$$\frac{dJ}{du} = \frac{2}{(t_p - t_0)} \int_{t_0}^{t_p} \left[f(\eta) - m^{\mathsf{T}} \cdot \left(\phi \cdot x_0 + \int_0^t \phi(t-\xi) \cdot g \cdot u(\xi) d\xi \right) \right] \left[- m^{\mathsf{T}} \cdot \left(\int_0^t \phi(t-\xi) \cdot g \cdot d\xi \right) \right] d\eta = 0 \tag{21}$$

We expanded equation 21 by substituting $u(\xi) = u_0 + \Delta u(\xi)$, in which u_0 is the starting steering control parameter and $\Delta u(\xi)$ is the change over the preview interval:

$$\begin{aligned}
\frac{dJ}{du} &= \int_{t_0}^{t_p} f(\eta) \left[- m^{\mathsf{T}} \cdot \left(\int_0^t \phi(t-\xi) \cdot g \cdot d\xi \right) \right] + \left[- m^{\mathsf{T}} \cdot \left(\int_0^t \phi(t-\xi) \cdot g \cdot d\xi \right) \right] + \left[- m^{\mathsf{T}} \cdot \phi(t-\xi) \cdot x_0 \right] \cdot \left[- m^{\mathsf{T}} \cdot \left(\int_0^t \phi(t-\xi) \cdot g \cdot d\xi \right) \right] \\
&+ (u_0 + \Delta u(\xi)) \left[- m^{\mathsf{T}} \cdot \left(\int_0^t \phi(t-\xi) \cdot g \cdot d\xi \right) \right] \left[- m^{\mathsf{T}} \cdot \left(\int_0^t \phi(t-\xi) \cdot g \cdot d\xi \right) \right] d\eta = 0 \\
\frac{dJ}{du} &= \int_{t_0}^{t_p} \left\{ f(\eta) + \left[- m^{\mathsf{T}} \cdot \phi(t-\xi) \cdot x_0 \right] + u_0 \left[- m^{\mathsf{T}} \cdot \left(\int_0^t \phi(t-\xi) \cdot g \cdot d\xi \right) \right] \right\} \left[- m^{\mathsf{T}} \cdot \left(\int_0^t \phi(t-\xi) \cdot g \cdot d\xi \right) \right] \\
&+ (\Delta u(\xi)) \left[- m^{\mathsf{T}} \cdot \left(\int_0^t \phi(t-\xi) \cdot g \cdot d\xi \right) \right]^2 d\eta = 0
\end{aligned} \tag{22}$$

Solving equation 22 for $u(\xi)$,

$$\Delta u(\xi) = \frac{\int_{t_0}^{t_p} \left\{ f(\eta) - m^T \cdot \phi(t - \xi) \cdot x_0 \right\} \left\{ m^T \int_0^t \phi(t_p - \xi) \cdot g \cdot d\xi \right\} d\eta}{\int_{t_0}^{t_p} \left[m^T \cdot \int_0^t \phi(t - \xi) \cdot g \, d\xi \right]^2 d\eta} \tag{23}$$

Equation 23 gives the basic algorithm for determining the change in steering command that minimizes the error between the desired vehicle lateral displacement at the end of the preview time interval and the predicted lateral displacement. The equations are written in terms of the fundamental matrix ϕ. Therefore, the next section will be developed in the way ϕ is calculated.

Expanding the fundamental matrix $\boldsymbol{\phi}$ gives the following:

$$\boldsymbol{\phi} = \begin{bmatrix} \begin{pmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{41} \end{pmatrix} \begin{pmatrix} x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \end{pmatrix} \begin{pmatrix} x_{13} \\ x_{23} \\ x_{33} \\ x_{43} \end{pmatrix} \begin{pmatrix} x_{14} \\ x_{24} \\ x_{34} \\ x_{44} \end{pmatrix} \end{bmatrix} \tag{24}$$

$$\boldsymbol{\phi}_0 = \begin{bmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{bmatrix} = [e_1 \; e_2 \; e_3 \; e_4] \tag{25}$$

Equation 25 is the initial condition for equation 24. Each column in equation 24 is an independent solution of the homogeneous part of equation 5, given below as equation 26.

$$\dot{x} = \boldsymbol{A} \cdot x = \boldsymbol{F} \cdot x \tag{26}$$

Each of the four independent solutions represented by the columns in equation 24 can be written as follows:

$$\begin{aligned} \dot{x}_1 &= F_{11}x_1 + F_{12}x_2 + F_{13}x_3 + F_{14}x_4 \\ \dot{x}_2 &= F_{21}x_1 + F_{22}x_2 + F_{23}x_3 + F_{24}x_4 \\ \dot{x}_3 &= F_{31}x_1 + F_{32}x_2 + F_{33}x_3 + F_{34}x_4 \\ \dot{x}_4 &= F_{41}x_1 + F_{42}x_2 + F_{43}x_3 + F_{44}x_4 \end{aligned} \tag{27}$$

With the definition of the matrix $\boldsymbol{F}$ given in equation 3, equation 27 can be rewritten as follows:

$$\begin{aligned} \dot{x}_1 &= x_2 + u \cdot x_4 \\ \dot{x}_2 &= \left[-2(C_{af} + C_{ar})/mU \right] x_2 + \left[2(bC_{ar} - aC_{af})/mU - U \right] x_3 \\ \dot{x}_3 &= \left[2(bC_{ar} - aC_{af})/IU \right] x_2 + \left[-2(a^2 C_{af} - b^2 C_{ar})/IU \right] x_3 \\ \dot{x}_4 &= x_3 \end{aligned} \tag{28}$$

A numerical integration of equation 28 is performed four times to produce the four independent solution vectors that comprise the columns of equation 24. With equation 25, the first numerical integration initializes (x_1, x_2, x_3, x_4) to e_1; the second initializes (x_1, x_2, x_3, x_4) to e_1; the third initializes (x_1, x_2, x_3, x_4) to e_3; and the fourth initializes (x_1, x_2, x_3, x_4) to e_4.

The time integral of the fundamental matrix in the denominator of equation 23 is also calculated with a numerical integration. Once the fundamental matrix and its time integral have been found by numerical integration, the matrix multiplications shown in equation 23 are performed. The numerator and denominator of equation 23 are formed and the division is performed. The results of equation 23 give the change in the steering command, and this is added to the previous steering command to give the steering command that minimizes the path error.

The elements of the $\boldsymbol{F}$ matrix given in equation 3 are calculated in subroutine "TRANS". The elements of the g vector are also calculated in "TRANS". "TRANS" also calculates the elements of the fundamental matrix $\boldsymbol{\phi}$ using equation 26 and a numerical integration. The calculation is performed four times—one time for each of the initial conditions given in equation 25. The time integration of the fundamental matrix

$$\int_0^\tau \boldsymbol{\phi} d\xi \xi$$

is also performed in "TRANS". The calculation of the fundamental matrix and its integral is performed over a set number of time intervals, and the result of each calculation is stored in an array. The array containing the stored fundamental matrices and their integrals is passed back to subroutine "NEWDRIVER".

The calculation of the change in steering command given in equation 23 is performed in subroutine "NEWDRIVER". We perform the matrix multiplication in equation 23 by calling subroutine "GMPRD". The integrations shown in the numerator and denominator are again done numerically. Finally, dividing the numerator by the denominator produces the change in steering command, which is added to the previous steering command to produce the current steering command. The steering command is passed back to subroutine "HUMAN_STEERING" which in turn passes the command back to "Interface_Steering". "Interface_Steering" passes the command onto the vehicle dynamics in the DADS model.

3. Three-Dimensional Driving Sensor Model

The 3-D driving sensor model provides information about the virtual environment to the steering model. The sensor model is attached to the vehicle dynamics model. The position and orientation of the vehicle determine the position and orientation of the sensor within the environment. The sensor does not change position or orientation relative to the vehicle, although

the code could be easily extended to model that type of sensor head motion. The sensor model produces a range map of the environment that is the basis of the planned or previewed path for the vehicle.

3.1 Definition of Three-Dimensional Obstacles

The environment for the driver model consists of 3-D objects. The objects are defined by a set of corner points and surfaces. Subsets of the corner points are used to identify the outer surfaces of the objects. Each type of 3-D object is defined only once but can appear in the environment any number of times. A unique location and orientation define each instance of the object type in the environment. The sensor model measures the distance from the sensor center to the surfaces of the 3-D objects in the environment. The range is measured along a series of scan rays. The distance to the closest object along this set of scan rays forms the range map of the environment.

The scan rays are generated on the basis of information read (scanned) by subroutine "HUMAN_DRIVER". The subroutine reads the maximum range at which the sensor can detect obstacles. The scanning pattern of the sensor is defined in terms of horizontal and vertical scan parameters. The scan angles are defined in terms of spherical coordinates centered on the sensor. Phi (ϕ) is the horizontal scan angle measured counterclockwise from the vehicle's X-axis. Theta (θ) is the vertical scan angle measured downward from the vehicle's Z-axis. "HUMAN_DRIVER" reads the maximum scan angle to the left and right of the X-axis and the maximum scan angle above and below the horizontal from a user-supplied input file. "HUMAN_DRIVER" also reads Delta_Theta and Delta_Phi which define the angular steps between scan rays. The scan pattern parameters are passed to "Interface_Steering" which, in turn, passes them to "Get_Obstacles".

The 3-D objects are initialized in subroutine "Get_Obstacles". Figures 3 and 4 show the process flow for defining the 3-D objects and placing them in the virtual environment surrounding the vehicle. The information is read only once, the first time the subroutine is called. The number of object types is read first. For each object type, the number of corner points is read. For each corner point in an object type, an X,Y,Z location is entered. The first set of corners is assumed to be part of a polygon in the Z = 0 plane. The corners are entered counterclockwise around the polygon, with as many as eight points describing the polygon. The second set of corner points forms a second polygon displaced in the negative Z direction. The corner points of the second polygon are again entered in a counterclockwise direction in a plane with a constant negative Z value.

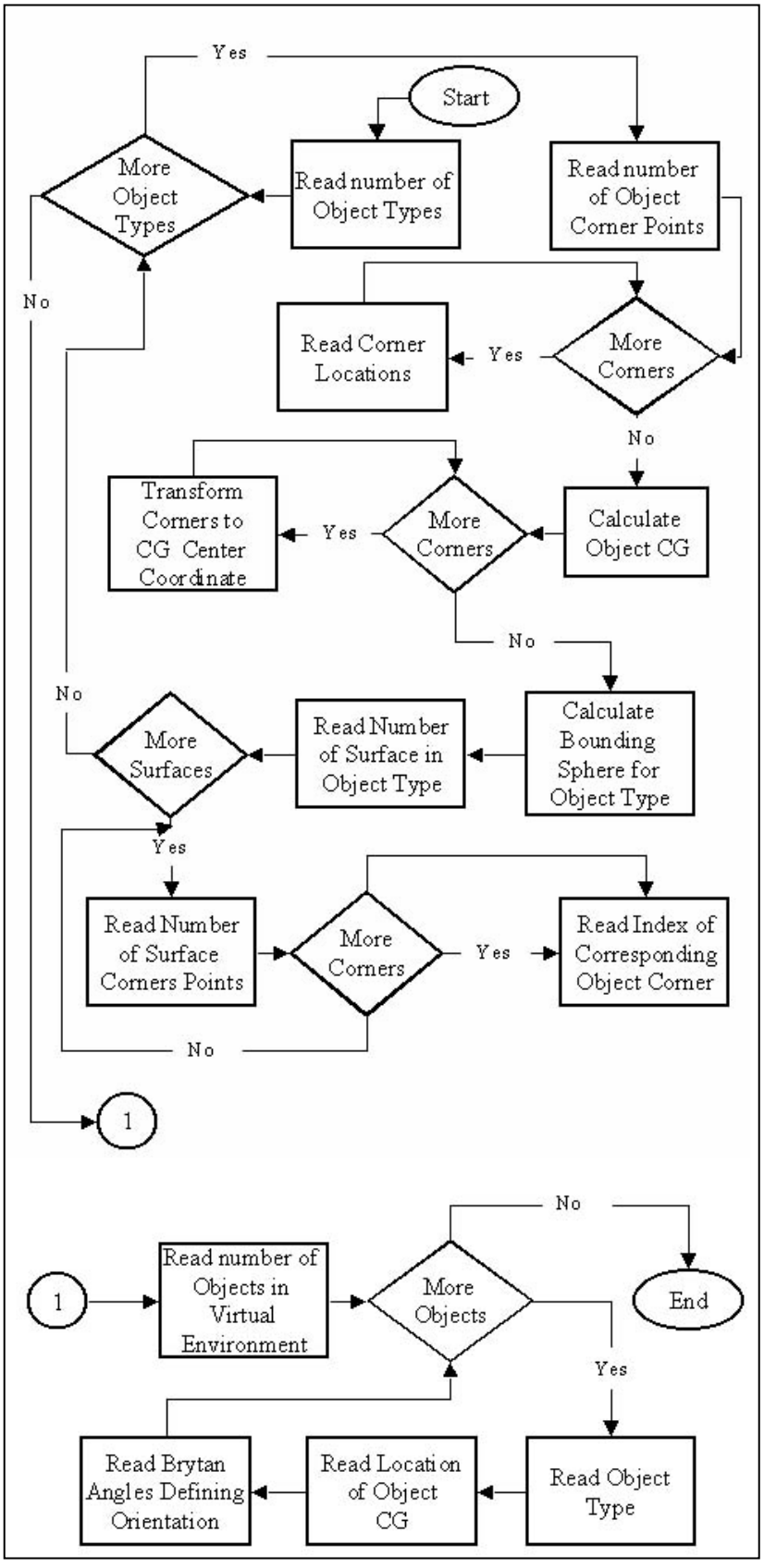

Figure 3. Process flow to define virtual environment.

Once the corner points have been read, the center of mass (CM) of the points is calculated. For the CM calculation, all the points are assigned a mass of one. The locations of the corner points are then transformed to a coordinate system centered at the CM. The transformed corner point locations are stored in a 2-D array. The first index in the array identifies the object type and the second the corner number within the object. The first index becomes the 3-D object identification number. The distance from the CM to each corner point is calculated and the largest distance is determined. A "bounding" sphere that encloses the object is then defined with its center at the CM and its radius equal to the distance from the CM to the farthest corner point.

Once the corner points have been defined, the data defining the surface of the 3-D object type are read. The number of surfaces in the object is read first. For each surface, the number of corner points in the surface is read. The surface corner points are a subset of the 3-D object corner points. The second index in the array of 3-D object corner points is used to identify surface corner point at the same location. These indices are stored in a 3-D array that defines the surface. The first index of the surface corner point array identifies the 3-D object, the second index identifies the surface, and the third identifies that corner within the surface. The second index becomes the surface ID number within the 3-D object type.

Once all the object types have been defined, the instances of the object types in the environment are read. First, the number of objects in the environment is read. For each 3-D object, the location of its CM in the inertial coordinate system is read. The locations of the object's CM are stored in a one-dimensional (1-D) array where the index indicates the order in which the instances of the object in the environment were read.

After the location of the objects CM has been entered, the orientation of the object is read. Next, the Bryant angles defining the 3-D object's orientation in the inertial coordinate system are read. The Bryant angles (ψ, θ, and ϕ) define a rotation about the X axis, followed by a rotation about the transformed Y axis and finally, a rotation about the twice-transformed Z axis. In other words, the object moves first in roll, then pitch, and finally in yaw. The Bryant angles are stored in 1-D arrays where the indices indicate the order in which the instances of the object in the environment were read.

Note that Bryant angles θ and ϕ are distinct from the θ and ϕ angles used to defined scan rays in spherical coordinates. Unfortunately, the standard definition for both Bryant angles and spherical coordinates use the same angle names, resulting in the overlapping definitions in this report.

3.2 Creating a Range Map of the Virtual Environment

Once the virtual environment has been defined in terms of 3-D objects, the sensor model can be used to create a range map for the steering model. The range map is created in subroutine "Get_Obstacles". "Get_Obstacles" is called by DADS every time step. DADS passes the vehicle position and orientation at the current time step to "Get_Obstacles." The subroutine passes back the range to the obstacles surrounding the vehicle along a set of pre-defined scan

rays for that time step. The process flow for creating a range map of the virtual environment is shown in figure 4.

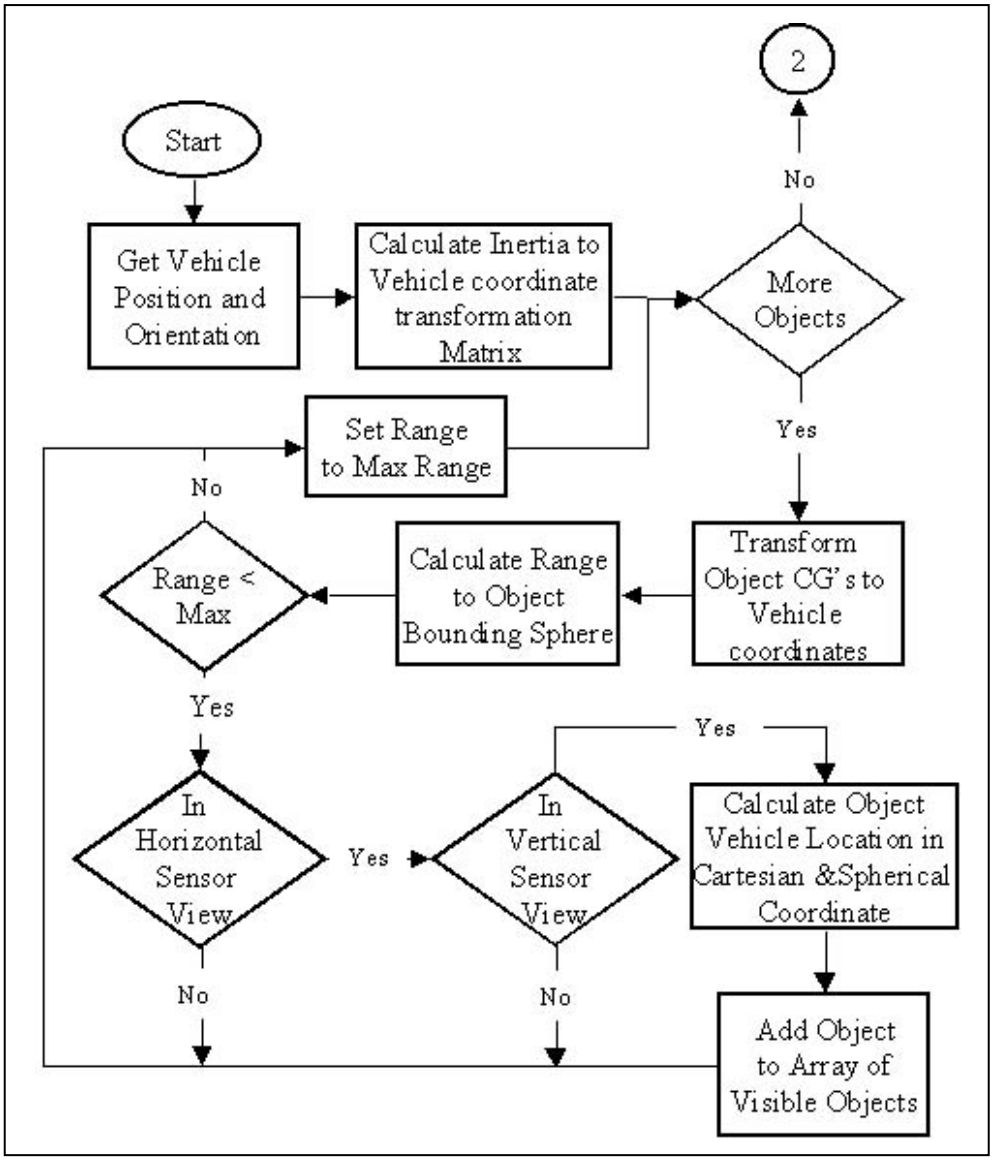

Figure 4. Process flow for range mapping.

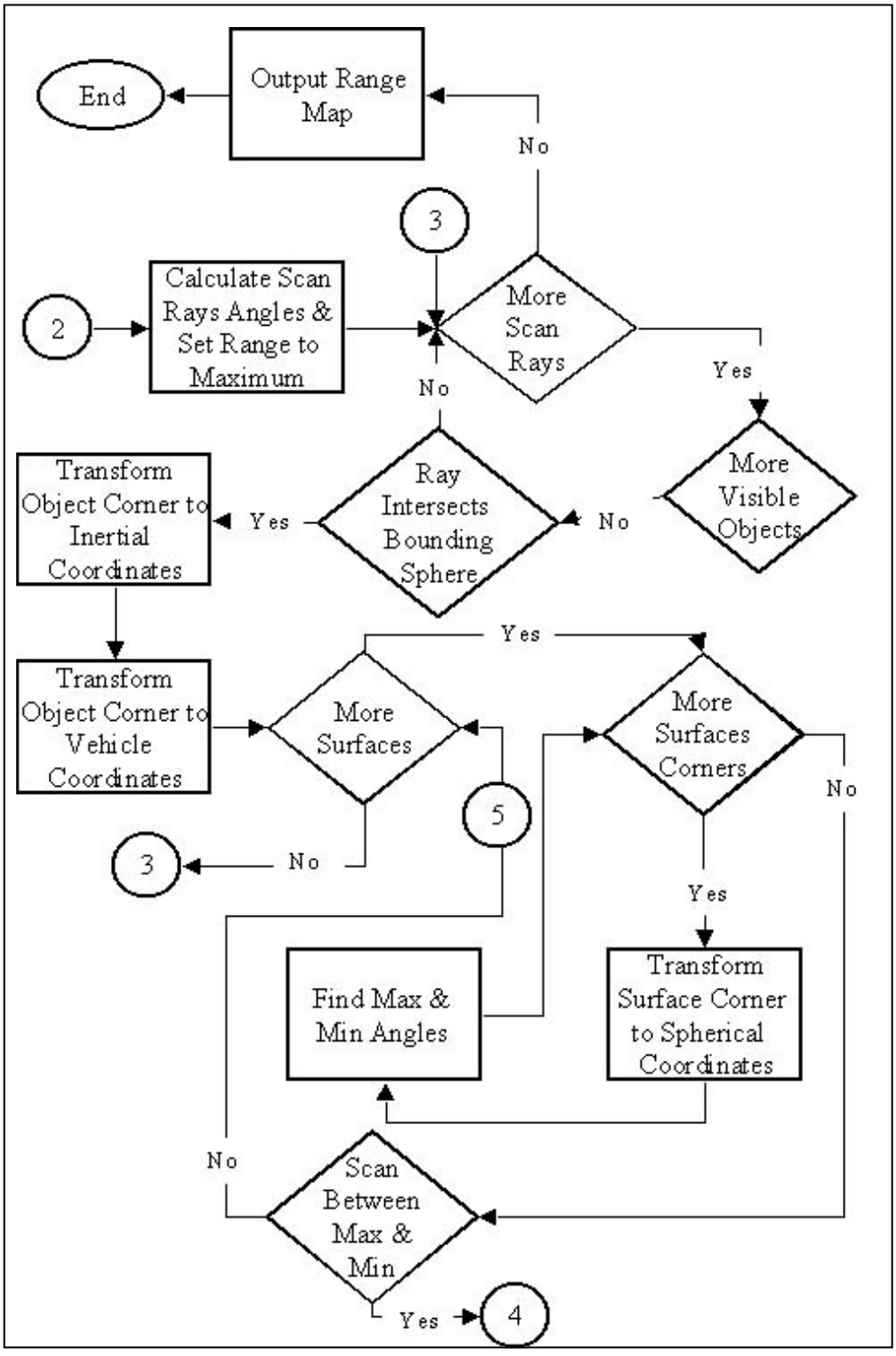

Figure 4. (continued)

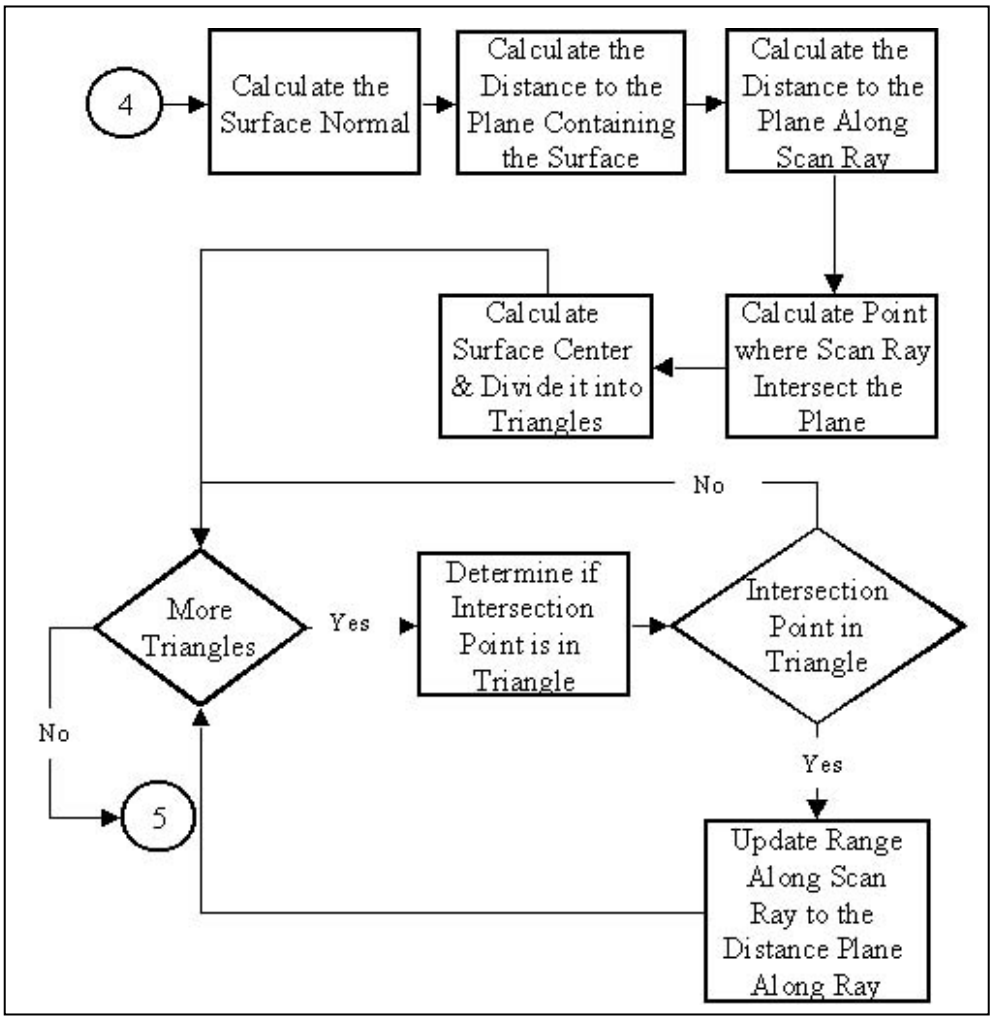

Figure 4. (continued)

The 3-D objects defined in the inertial coordinate system are transformed to the vehicle-centered coordinate system. The first step in this process is to calculate the transformation matrix that rotates coordinates in the inertial frame into the vehicle coordinate system. The transformation matrix required is calculated from the set of three Bryant angles defining the current orientation of the vehicle. Elements of the three by three transformation matrix are calculated by subroutine "BRYANT_MATRIX" which is called by "Get_Obstacles".

The transformation matrix, along with information about the 3-D objects, is passed into subroutine "VEH_COORDS" which is called by "Get_Obstacles". "VEH_COORDS" first transforms the CM location of all the 3-D objects to vehicle-centered coordinates. The coordinate transformation is performed by subroutine "TRANSFORMER," which is called by "VEH_COORDS". "TRANSFORMER" uses the transformation matrix and the current vehicle position to rotate and displace the CM location inertial coordinates into vehicle coordinates.

Once the object CM locations have been transformed to vehicle coordinates, "VEH_COORDS" checks the range to each object. Each object is checked to see if any part of its bounding sphere, centered at the CM, is within the maximum sensor range of the vehicle. If part of the bounding sphere falls within the maximum sensor range, then a check is made to see if any part of the bounding sphere falls within the angular range of the horizontal scan of the sensor. If some part of the bounding sphere falls within the horizontal scan range, a final check is made to see if part also falls within the vertical angular scan range. Those 3-D objects that have at least part of their bounding sphere within the scan pattern of the sensor are stored in a new array. The array contains the 3-D object identification number, object type, range to the object, the object location in vehicle Cartesian coordinates and in vehicle spherical coordinates.

Once "VEH_COORDS" returns to "Get_Obstacles," that subroutine calls subroutine "SCAN" and passes the array of objects with bounding spheres within the sensor scan pattern. "Scan" walks through the set of scan rays, starting from the upper left. The subroutine first marches the scan rays horizontally left to right in steps of "Delta_Phi". When a horizontal scan is complete, the vertical scan angle is decreased by "Delta_Theta" and the next complete horizontal scan is initiated. The pattern is repeated until the lowest horizontal scan is completed. The range along all scan rays is initially set to the maximum sensor range. The range is decreased only if, later in the process, the scan ray is found to intercept an object surface.

For each scan ray, a check is performed against each object within the sensor scan pattern to see if the ray intersects any part of the bounding sphere. If the scan ray does not intersect the bounding sphere of an object, then the range along that ray remains set to the maximum sensor range. If the scan ray intersects the bounding sphere of a 3-D object, then subroutine "SCAN_SURF" is called to see if it intersects a surface of the object.

"SCAN_SURF" is used to determine if a scan ray intersects any of the external surfaces of a 3-D object. First, the corner points of the object to be examined are transformed to inertial coordinates from object-centered coordinates via the subroutine "TRANSFORMER". The corner points are then transformed from inertial to vehicle coordinates, again with "TRANSFORMER". "SCAN_SURF" then marches through the external surface of the 3-D object in vehicle coordinates to see if the scan ray passed through one of the surfaces.

The corner points of the surface are transformed into vehicle spherical coordinates. The corner points with the maximum and minimum horizontal angle, ϕ, are found. The corner points with the maximum and minimum vertical angle, θ, are also identified. If the scan ray's θ and ϕ values do not fall between the maximum and minimum for the surface, it missed the surface. If it does fall with the maximum and minimum, it may strike the surface. For those rays, we perform a further check by calling subroutine "RAY_INTERSECT".

Subroutine "RAY_INTERSECT" first calculates the unit surface normal for the surface in question. "RAY_INTERSECT" then calculates the distance from the sensor to the plane containing the surface and the distance from the sensor to the plane along the scan ray. It uses

this information to calculate the point where the scan ray intersects the plane containing the surface. Once the intersection of the scan ray and the plane has been identified, it must be determined if that point lies inside the actual surface.

First, the center of the surface is determined and then, the surface is divided into triangles. Each triangle has the surface center as one vertex and two surface corner points as the other vertices. The subroutine then marches through the triangles one at a time, checking to see if the intersection point lies within it. The algorithm used to check the triangles was taken from *Graphic Gems* (*4*). If the intersection point does not lie in any of the triangles, the scan ray missed the surface and the range remains set to the maximum sensor range. If the intersection point is in one of the triangles, then the range is set to the distance from the sensor to the plane, as calculated before. Subroutine "RAY_INTERSECT" returns the range to subroutine "SCAN_SURF".

In "SCAN_SURF," each surface in the object is checked. If the scan ray intersects more than one surface, the range to the object is set equal to the shortest range to any of the surfaces in the object. "SCAN_SURF" returns the range to the object to subroutine "SCAN" where the range for that scan ray is stored in an array of ranges for all scan rays. The range array is 2-D, with the first index identifying the vertical scan angle and the second the horizontal scan angle. The range array, along with the array containing the horizontal scan angle, ϕ, and the vertical scan angle, θ, form a range map of the environment around the vehicle in spherical coordinates. The range map is returned by "SCAN" to subroutine "Get_Obstacles". "Get_Obstacles" returns the range map to "Interface_Steering" which passes it onto the steering model.

4. Obstacle Avoidance and the Previewed Path

The range map returned by "Get_Obstacles" is 3-D. The obstacle avoidance algorithm and steering algorithm taken from reference (*1*) can only handle 2-D range maps. The 3-D map is reduced to a 2-D map within "Interface_Steering" by the sorting of each vertical slice of the scan array and by the selection of the minimum range in that slice. The minimum range in the vertical slice is stored in the "RD" array. The horizontal angle "ϕ" associated with each vertical slice is stored in the array "TH". "RD" and "TH" are passed to subroutine "Avoid_Obstacles" along with the position and yaw angle of the vehicle.

Subroutine "Avoid_Obstacles" calls the subroutine "CALSRS," which calculates the average range over the "RD" array and stores it in the variable "RSTAR". The subroutine "CALCTH" is then called and calculates the average of the product of the range and the angle over the arrays "RD" and "TH". The average is stored in the variable "THSTAR". Finally, "Avoid_Obstacles" calls "CALCTH," which calculates the range within the 2-D range map along the "THSTAR"

angle. If "RSTAR" is greater than the range along "THSTAR," then "RSTAR" is reduced by 20%.

"THSTAR" and "RSTAR" define a point on the preview path in terms of vehicle-centered polar coordinates. The same point in the previewed path in inertial coordinates is then calculated and stored in "XSTAR" and "YSTAR". "XSTAR" and "YSTAR" define the point in the path that vehicle should occupy at the end of the preview time.

"Avoid_Obstacles" returns "XSTAR" and "YSTAR" to "Interface_Steering". "Interface_ Steering" calls "HUMAN_STEERING," which adds the last "XSTAR" and "YSTAR" values to the previewed path. The previewed path is used to determine the future route of the vehicle. The algorithm for "HUMAN_STEERING" was covered in detail in section 2.

5. Interfacing the Human Steering and Vehicle Dynamics Models

The interface between the human steering model and the vehicle dynamics model is accomplished with a series of FORTRAN[2] subroutines. The external link into the DADS vehicle simulation is via the user-defined force/torque subroutine "FR3512". "FR3512" provides access to the UserAlgorithm control element in which a control node is defined to allow steering control torques to be applied to the steering actuator model. Movement of the steering actuator components, which is attributable to the control torques, causes an angular change in the vehicle's steered wheels relative to the vehicle chassis, which in turn causes the vehicle to change direction. Vehicle state information and vehicle parameter data are also collected in subroutine "FR3512" when control input nodes are accessed. The control element input nodes within the DADS vehicle model act as sensors for collecting vehicle state and parameter data. The vehicle data are passed from "FR3512" to subroutine "VEH_STEER". Within "VEH_STEER," vehicle data undergo unit conversions before being passed to subroutine "INTERFACE_STEERING". Further, "VEH_STEER" receives steering commands from "INTERFACE_STEERING" in the form of steering angles and then converts these commands to steering motor torques. The commanded steering torque is then passed back to "FR3512". "INTERFACE_STEERING" is the entrance into the human steering model. First, a call is made to subroutine "HUMAN_DRIVER" to collect driver model parameters; next, a call is made to subroutine "GET_OBSTACLES" to retrieve the obstacle data information in the form of a range map. The range map data are used in the call to "AVOID_OBSTACLES," which generates a path for the vehicle to follow to avoid the obstacles. "AVOID_OBSTACLES" returns a location, XSTAR and YSTAR, for the human steering model to steer the vehicle toward.

[2]Formula Translator

XSTAR and YSTAR are passed to subroutine "HUMAN_STEERING" which generates and returns a commanded steering angle (DFW[3]).

5.1 Steering Commands and Steering Motor Torque Control

The steering command, which originates as a DFW within subroutine "HUMAN_STEERING," is a torque that is applied to the steering actuator model within the DADS vehicle simulation. DFW is passed from subroutine "HUMAN_STEERING" via "INTERFACE_STEERING" to subroutine "VEH_STEER" where it is converted from a commanded steering angle to a steering motor torque, STRCOM. "VEH_STEER" employs two separate algorithms for converting steering angles to steering motor torques, each independently applied, depending on the steering system being modeled. The first is a simple proportional gain function wherein the output steering motor torque is linearly proportional to the difference between the DFW and the present steered wheel angle (RFWANG) multiplied by the steering gain coefficient (STGAIN). RFWANG is the angle of the steered wheels with respect to the vehicle chassis. RFWANG is typically the average of the right and left steered wheels when an Ackermann steering system is employed. An Ackermann system uses a steering linkage geometry in which the right and left wheels will turn to different angles, depending on the radius of the arc that each wheel traverses. The basic proportional steering motor torque function is shown in equation 29.

$$\mathrm{STRCOM} = (\mathrm{DFW} - \mathrm{RFWANG}) \cdot \mathrm{STGAIN} \tag{29}$$

The second or alternate algorithm is a velocity-dependent nonlinear decreasing gain function wherein the output steering motor torque is linearly proportional to the difference between the DFW and the RFWANG multiplied by a nonlinear steering gain function. The STGAIN is modified by a velocity-dependent decreasing parabolic function based on the vehicle chassis longitudinal velocity and a velocity gain-limiting coefficient (XDGLIM). The effect of this nonlinear gain function on the steering system is to increase the applied steering motor torque while the vehicle is at rest or during low speed travel. The increased steering motor torque is needed to offset the increased resistance to turning the steered wheels while at rest or during low vehicle speeds. The velocity-dependent steering motor torque function is shown in equation 30.

$$\mathrm{STRCOM} = (\mathrm{DFW} - \mathrm{RFWANG}) \cdot \mathrm{STGAIN} \cdot \left(2 - \left(\mathrm{XD} / \mathrm{XDGLIM}\right)^2\right) \tag{30}$$

in which XD is the vehicle chassis longitudinal velocity.

The steering motor torque (STRCOM) is then passed back to subroutine "FR3512" where it is linked to the output node for the UserAlgorithm Control element named FR_STEER_ACT_COMMAND. The FR_STEER_ACT_COMMAND node within the DADS simulation control architecture is further linked to the Control Output One-Body element named FR_STEER_ACT_TORQ. STRCOM is applied as a ZL.TORQUE to the ROTARY_STEER_ACT body. The

[3] not an acronym

ROTARY_STEER_ACT body is the main component of the steering motor model within the vehicle model. The steering motor torque is applied to the local z-axis of the cg triad of the ROTARY_STEER_ACT body, which causes the steering actuator to rotate about a revolute joint[4] connected to the vehicle chassis.

The rotary steering actuator body has an extended lever arm, perpendicular to its local z-axis and acting as a simulated pitman arm (i.e., connecting rod), to convert the rotary steering motion into a translational steering motion. A DRAG_LINK body is connected via spherical joints between the extended lever of the rotary steering actuator and the steering RACK. A spherical joint connects two component bodies in which relative rotary motion is allowed about all axes and relative translational motion is not permitted along any axis. The drag link body transfers the translational steering motion into a lateral movement of the steering rack body. The steering rack body is connected via universal joints to a left and a right tie rod body, named, respectively, LF_TIE_ROD and RF_TIE_ROD. A universal joint connects two component bodies in which relative rotational motion is allowed about two axes and relative translational motion is not allowed along any axis. The individual left and right tie rod bodies transfer the lateral steering rack motion to their respective wheel hub bodies, LF_WHEEL_HUB and RF_WHEEL_HUB. Each wheel hub body has a lever arm extended perpendicular to its steering axis; this arm acts as a simulated steering knuckle. The tie rod bodies are connected via spherical joints to their respective steering knuckles. These connections allow the lateral steering motion to be converted into angular motion about the left and right steering axes.

Wheel bodies, LF_WHEEL and RF_WHEEL, are connected to their respective wheel hub bodies by revolute joints. These joint connections allow the wheel bodies to rotate relative to the vehicle chassis. The wheel bodies form the basis for the individual tire models that generate the forces for displacing the vehicle chassis. Lateral displacement of the vehicle chassis is accomplished by the generation of lateral forces within the tire models. The lateral forces for turning, known as cornering forces, are created when the rolling wheel bodies have an angular displacement about their steering axes, relative to the vehicle chassis. This angular displacement of the wheel body creates a slip angle within the rolling tire model. Slip angle is defined as the difference between the steered angle of the tire and the actual angle of tire travel relative to the vehicle chassis. Slip angle is attributable to the deformation or twisting of the tire carcass during cornering, which produces a resultant lateral force. This lateral force is transmitted from the tire model through the wheel body to the vehicle chassis, which results in the lateral displacement or turning of the vehicle.

[4]A revolute joint is a connection between two component bodies in which rotary motion is permitted about a single axis (typically the z-axis), and relative translational motion is not permitted along any axis.

5.2 Vehicle Speed and Drive Torque Control

The DADS vehicle velocity during simulation runs is controlled through a UserAlgorithm Control element that provides a control node to the user-defined force/torque subroutine "FR3512". The UserAlgorithm Control element, SPEED_CONTROL, has an output node known as DRIVE_TORQUE. DRIVE_TORQUE is accessed through subroutine "FR3512," which calls the velocity-controlling subroutine "SPEED". The control node information RR_WHEEL_RATE, which provides the angular rate of the right rear wheel, is passed to subroutine speed. The algorithm within subroutine SPEED is a simple velocity controller in which driving and braking torques are generated, based on the angular rate of the simulated vehicle's right rear wheel. A relatively constant vehicle velocity is attained by the generation of a drive torque output that is proportional to the angular rate error of the right rear wheel. First, a desired angular wheel rate is set; then, the desired wheel rate is subtracted from the actual rate of the right rear wheel. The differencing creates an angular rate error, which is then multiplied by a torque gain constant. The resulting torque output is then limited to avoid generating excessive braking or driving torques. The output drive torque is then passed back to the output control node, DRIVE_TORQUE, via subroutine "FR3512". The control node, DRIVE_TORQUE, is linked to each of four Control Output One-Body elements. The Control Output One-Body elements, RR_WHEEL_TORQUE, LR_WHEEL_TORQUE, RF_WHEEL_TORQUE, and LF_WHEEL_TORQUE, represent drive and braking torque being distributed to each of the vehicle's four wheels.

5.3 Vehicle State and Data Parameters

The vehicle state and data parameters are collected within the DADS vehicle simulation and then communicated to the human driver model through subroutine "FR3512". The vehicle state and parameter information is sensed and collected from the vehicle dynamics model in the form of control node information and rigid body array data. The DADS global coordinates are equivalent to the inertial coordinates in the human steering model. The control node information is

CHASSIS_X_POS - the global longitudinal position of the vehicle chassis
CHASSIS_Y_POS - the global lateral position of the vehicle chassis
CHASSIS_Z_POS - the global vertical position of the vehicle chassis
CHASSIS_X_VEL - the global longitudinal velocity of the vehicle chassis
CHASSIS_Y_VEL - the global lateral velocity of the vehicle chassis
CHASSIS_Z_VEL - the global vertical velocity of the vehicle chassis
CHASSIS_ROLL_ANG - the roll angle of the vehicle chassis
CHASSIS_PITCH_ANG – the pitch angle of the vehicle chassis
CHASSIS_YAW_ANG – the yaw angle of the vehicle chassis
CHASSIS_ROLL_RATE – the roll rate of the vehicle chassis
CHASSIS_PITCH_RATE – the pitch rate of the vehicle chassis
CHASSIS_YAW_RATE – the yaw rate of the vehicle chassis
CHASSIS_LOCAL_Y_VEL – the local lateral velocity of the vehicle chassis
FR_STEER_ACT_SENSOR – the angle of the rotary steering actuator with respect to the

vehicle chassis
FR_WHEEL_WRT_CHASSIS – the angle of the front wheel with respect to the vehicle chassis
RF_WHEEL_RATE – the angular rate of the right front wheel
RR_WHEEL_RATE – the angular rate of the right rear wheel

Rigid body array data are

NUMWHL – number of wheels on the vehicle
NUMRB – number of rigid bodies within the vehicle simulation
CRSTFR(1) – cornering stiffness of the right front wheel
CRSTFR(2) - cornering stiffness of the right rear wheel
CRSTFL(1) - cornering stiffness of the left front wheel
CRSTFL(2) - cornering stiffness of the left rear wheel
VMASS – mass of the vehicle
IZZ5 – yaw moment of inertia of the vehicle
ZFR(1) – load on the right front tire
ZFR(2) – load on the right rear tire
ZFL(1) – load on the left front tire
ZFL(2) – load on the left rear tire

Control node information is collected within the vehicle simulation through the use of Control Output One-Body elements and Control Output Two-Body elements. We access the Control Output One-Body element, CHASSIS_PITCH_ANG, for example, by specifying the simulation body of interest, CHASSIS_SENSOR, the body's triad where the sensing will take place, CHASSIS_SENSOR_CG, and the type of data to collect from this location. For this example, to collect the chassis pitch angle, the information collected was the second Bryant angle. A triad is defined within the simulation architecture as a 3-D coordinate origin. We access the Control Output Two-Body element, FR_STEER_ACT_SENSOR, for example, by specifying the first simulation body of interest, CHASSIS, the first body's triad, ROT_STR_ACT_MNT, the second simulation body of interest, ROTARY_STEER_ACT, the second body's triad, ROTARY_STEER_ACT, and the type of data to be collected. The information collected in this example is the angular difference about the z-axes of the triads of the two bodies. Control node information, such as the two examples given, is then passed from the DADS simulation by a GETNODE function call within the user-defined subroutine "FR3512". This information is then passed to subroutine "VEH_STEER" for use within the human driver model.

Rigid body array data are collected directly through subroutine "FR3512" by a call to the function INDXAR. This function returns the index of the desired data element from the array of interest. This index is then used to return the desired value from the array. The arrays that may be searched via the INDXAR function are

A – all real data for the system
IA – all integer data for the system
Q – global generalized coordinates
QD – global generalized velocities

QDD – global generalized accelerations
FRC – global generalized forces

To find the index of the desired element, the function call to INDXAR must include the name of the array being searched, the dimension of the solution (2-D or 3-D), the module number of the element, the instance of the element, the index of the element array, and the "IA" array. For example, for us to acquire the value for ZFR(1), the normal force on the right front tire, the following must be specified in the call to INDXAR:

ZFR(1) = A(INDXAR('A',3,31,3,42,IA))

in which

A – the name of the real data array
3 – indicates a 3-D solution
31 – tire force element module number
3 – the instance of the tire element (the right front tire is the third instantiation of the tire element)
42 – the index of the normal force in the tire element
IA – the name of the integer data array

This value, along with the other values acquired from the rigid body array, are then passed from subroutine "FR3512" to subroutine "VEH_STEER". The rigid body array data and the control node information are then passed to subroutine "INTERFACE_STEERING" for use within the human driver model.

6. Vehicle Dynamics Model

6.1 High Mobility Multipurpose Wheeled Vehicle (HMMWV) Model

The DADS vehicle dynamics model chosen for interfacing to the human driver model was a model of the HMMWV that was developed by the University of Iowa's Center for Virtual Proving Ground Simulation. A HMMWV model was selected since it would likely enable a more direct comparison with the original driver model DADS interfacing results. The DADS HMMWV, developed by the University of Iowa, is an advanced 22-body model that includes the ability to simulate propulsion, braking, and steering. During development of the interface between the human driver model and the DADS vehicle dynamics model, a surrogate DADS vehicle model was employed. This vehicle model contained a rotary, hydraulic steering system, and the development of the steering control algorithms focused upon this type of steering system. When the human driver model was interfaced to the University of Iowa-developed HMMWV, a

few modifications of the model were required to enable the use of the rotary steering controller. Further, the HMMWV model required the addition of a number of control architecture elements.

6.2 Steering Control Modifications

The HMMWV, as obtained from University of Iowa, employs a "rack and pinion" type steering system, in which the steering rack is driven laterally by a translational actuator. The ends of the steering rack are attached to tie rods at a joint, which transmit the lateral translational motion to the steering knuckles. The steering knuckles, which mount the wheel hubs and wheels, convert the translational motion to angular motion for steering the wheels. The human driver model employs a rotary steering actuator control algorithm; thus, a rotary steering actuator model needed to be added to the HMMWV model. The modification required disabling the HMMWV model's present translational actuator and the fitting of a rotary actuator and required linkage. A rotary steering actuator body, ROTARY_STEER_ACT, was connected to the HMMWV chassis by a revolute joint. The rotary steering actuator body has an extended lever, acting as a pitman arm. A drag link body, DRAG_LINK, is connected to the pitman arm via a spherical joint. The drag link body uses another spherical joint at its other end to connect to the steering rack body, RACK. This actuator and linkage model convert the rotary steering input into the lateral translational motion required by the HMMWV model. The rotary steering input, STRCOM, is the commanded steering motor torque from the vehicle steering interface model, "VEH_STEER". To properly control the steering torque, a DADS friction element was developed and included in the steering motor model. The friction element acts as a damper in the feedback steering control system. The friction element, ROT_STEER_ACT_FRICT, takes the form of bearing friction within the rotary steering actuator body and is applied to the revolute joint, REV_ROTSTR_ACT_CHASS, that connects the rotary steering actuator to the vehicle chassis. The bearing friction is based on the following specifications:

bearing radius – 2.0 inches

bearing height – 4.0 inches

static coefficient of friction – 0.2

dynamic coefficient of friction – 0.15

linear velocity threshold – 0.001 inch/second

The frictional damper is necessary within the steering motor model to alleviate steering system control jitter during transitional steering commands.

6.3 Control Elements for Vehicle State Sensing and Commanded Actuation

Control element additions are required in the DADS vehicle model for the human driver to receive vehicle state information and to output driving commands. A series of Control Input

One-Body and Control Input Two-Body elements was added to the University of Iowa HMMWV model. A comprehensive listing of the control input elements is included in section 4.3. These control input elements function as virtual sensors to acquire the present vehicle state during a simulation, which is required by the human driver steering controller. Control Output One-Body elements were added to the vehicle model to function as virtual actuators controlled by the human driver model. The control output elements are

FR_STEER_ACT_TORQ - for actuating the rotary steering actuator

RR_WHEEL_TORQUE - for powering the right rear wheel

LR_WHEEL_TORQUE – for powering the left rear wheel

RF_WHEEL_TORQUE – for powering the right front wheel

LF_WHEEL_TORQUE – for powering the left front wheel

These control output elements allow the human driver model to apply control torques to the vehicle dynamics model during a simulation run to alter its position and velocity.

7. Validation of HMMWV Results

Figure 5 shows a view of the slalom course and the HMMWV model. The slalom course consists of a single lane entry, a two-lane section with two square obstacles, and a single lane exit. The boundaries of the course are lined with "Jersey" barriers. The first obstacle abuts the left-hand Jersey barrier boundary. The second obstacle abuts the right-hand Jersey barrier boundary. Figure 6 shows a series of frames of the HMMWV model maneuvering through the slalom course.

The slalom course has the same geometry in the X-Y plane as the course described in reference (*1*). The entry and exit lanes are 12 feet wide and 175 feet long. The two-lane section is 24 feet wide and 324 feet long. The two square obstacles measure 12 feet on each side. The first obstacle is 100 feet from the start of the two-lane section. The second obstacle is 212 feet from the start of the two-lane section.

Figure 7 shows the results of a test run of the HMMWV model through the slalom course. The neuromuscular delay and "look-ahead" time used are the same as those in reference (*1*). The HMMWV accelerates in the entry lane and reaches 40 mph as the vehicle enters the two-lane section. The vehicle then maintains 40 mph for the rest of the course. This is the speed used for similar runs in reference (*1*).

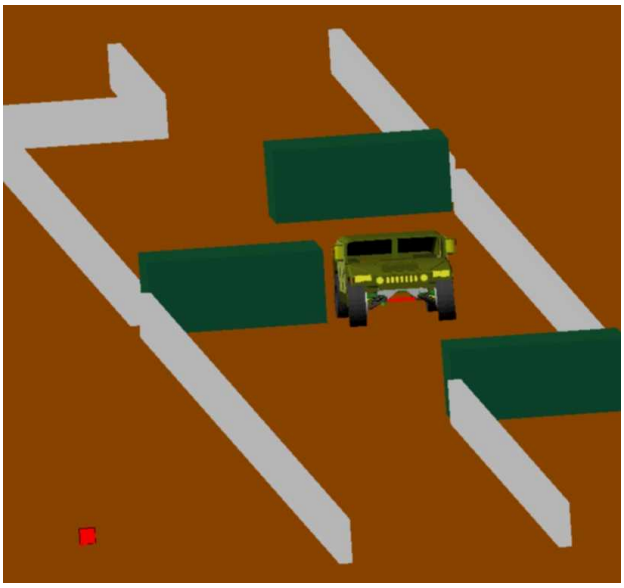

Figure 5. HMMWV in slalom track.

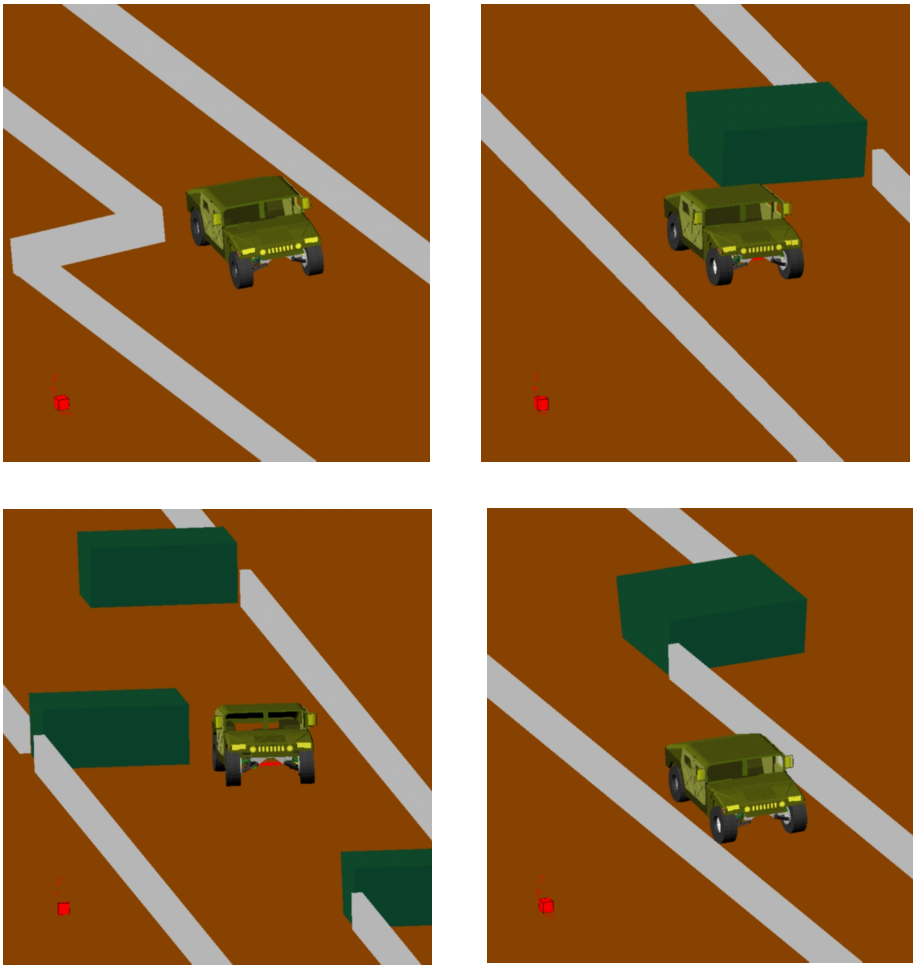

Figure 6. HMMWV maneuvering through the slalom course.

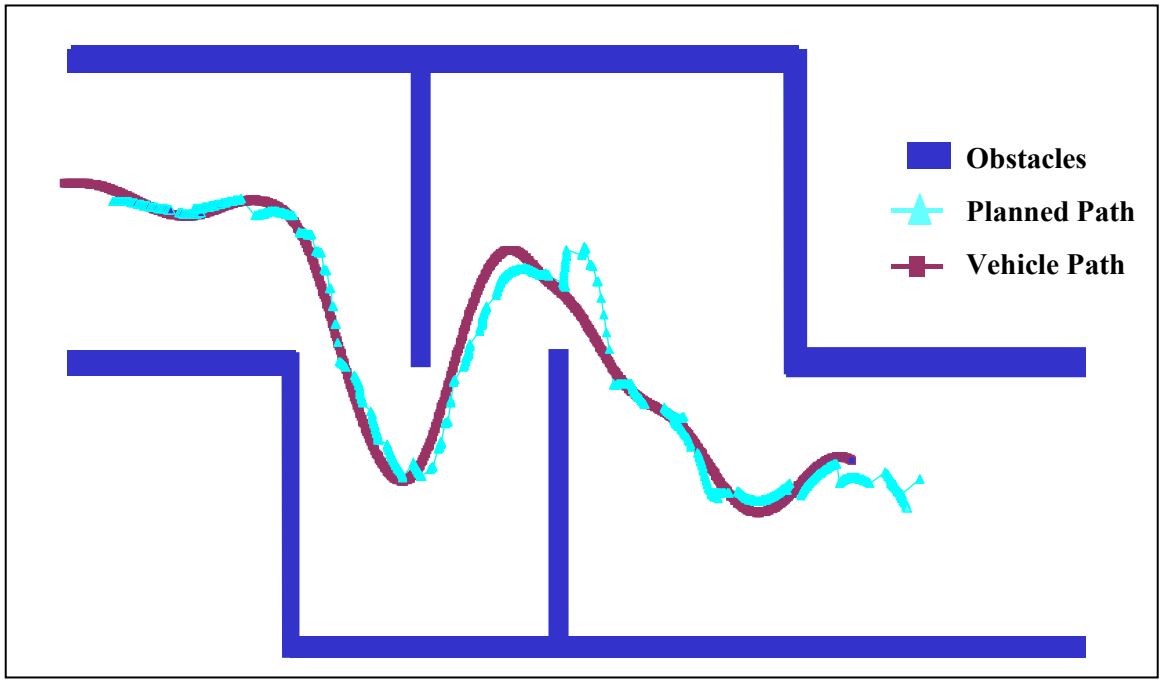

Figure 7. Planned and actual vehicle path through slalom course.

In Figure 7, the heavy dark blue lines are the Jersey barriers defining the course boundaries. The light blue line is the planned or previewed path. Each of the diamond symbols indicates an "XSTAR,YSTAR" pair generated by subroutine "Avoid_Obstacles". The dark purple line is the path followed by the vehicle's center of mass. Each of the squares is the vehicle's position as reported by the DADS simulation model.

Figure 8 shows a comparison between the steering model developed in this report, the existing steering model, and experimental data in reference (*1*). The slalom course, vehicle type, and speed driven are the same in all three cases. The red line in the figure is the path of the vehicle's cg calculated by the model described in this report. The heavy black line is the model runs from reference (*1*). The thin black lines are experimental runs performed by human drivers as reported in reference (*1*). The red path lies close to or within the set of experimental paths, thus validating that the current model does a reasonable job of simulating human steering for this vehicle on this course.

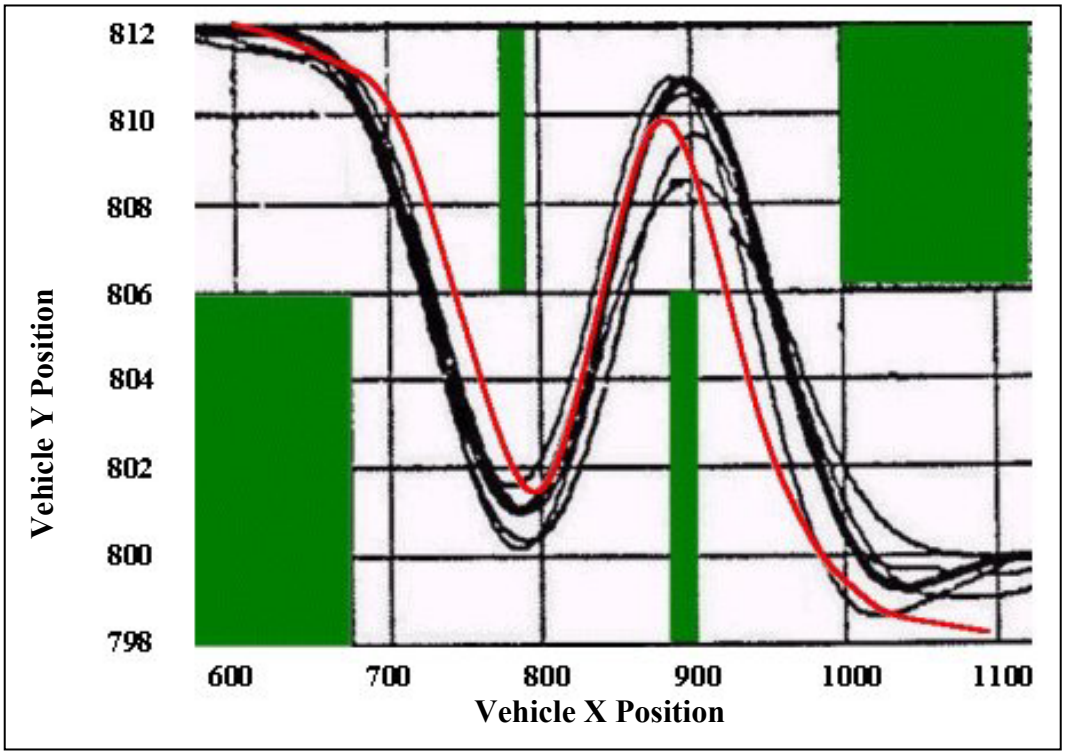

Figure 8. Comparison of steering models and experimental results.

8. Conclusions

The steering model described in this report successfully simulates a human driving a vehicle through a series of 3-D obstacles. The model simulates the solid geometry of a range finding, driving sensor attached to the vehicle and operating in a simple 3-D virtual environment. The model also simulates a control loop representing a human steering, with neuromuscular delay and look-ahead time. The sensor model and the steering model are coupled to a detailed vehicle model generated in the commercial off-the-shelf (COTS) simulation environment DADS.

If a human in the loop is used to provide steering commands to the vehicle dynamics model, then the model must be simplified to run in real time. If a predetermined series of steering commands is used, these commands will not reflect the feedback that occurs between the steering commands and vehicle dynamics. This model allows for complex vehicle tests to be simulated while the feedback is maintained between the steering commands and the vehicle dynamics but without a human in the loop. In this way, complex tests can be simulated efficiently on small computers at slower than real time. Running the simulation in the background can provide input to parametric studies. These studies estimate which candidate vehicle designs will perform best on a standard test course. The results could be used to guide vehicle design or for better planning of physical experiments.

9. References

1. MacAdam, C. C. *Development of Driver/Vehicle Steering Interaction Model for Dynamics Analysis*; U.S. Army Tank-Automotive Command, Research, Development & Engineering Center: Warren, MI, December 1988.
2. Boyce, W. E.; DiPrima, R. C. *Elementary Differential Equations and Boundary Value Problems*; 2nd Edition; John Wiley and Sons, Inc., New York, 1969.
3. Butkov, E. *Mathematical Physics*; Andison-Wesley Publishing Company, Reading, Massachusetts, 1968.
4. Glassner, A. S. *Graphic Gems*; Academic Press Inc., San Diego, CA, 1990.

INTENTIONALLY LEFT BLANK.

Appendix A. Code Makefile

```
        *****************
        * CODE MAKEFILE *
        *****************

#CC = CC
#CFLAGS = -g -strictIEEE -static

#NOTE: Use SGI make command /usr/bin/make

##----------- IRIX defs-------------------------
CXX = CC
CFLAGS =-g -O -D_POSIX_PTHREAD_SEMANTICS \
      -I/other/OTB.rp/include/global \
        -I/other/OTB.rp/include/libinc/ \
        -I/other/OTB.rp/include/gnuinc \
        -I/other/OTB.rp/libsrc/libctdb/ \
        -I/other/OTB.rp/libsrc/libctdb/compiler/ \
        -I/other/OTB.rp/

LDFLAGS2=  -g -lm  -L/other/OTB.rp/lib
LIBS2= -lctdb -lcoordinates -lgeometry -lgcs -lconstants -lvecmat -lreader -
lgeo3d -lbasic -lm

FTNLIBS=-L /usr/lib/DCC -lc -lftn
#CCLIBS= -L /usr/lib/CC  -I /usr/include

.SUFFIXES: .o .f .bd

DADSLIBDIR =  /e3a/dads95/dadslib/

MEXLIBDIR =  /e3a/dads95/dadslib/sgi64/

LMGRLIB =  liblmgr.a

DADSLIBSMEX= \
        ${MEXLIBDIR}blockda.o \
        ${MEXLIBDIR}revbd.o \
        ${MEXLIBDIR}xdummy.o \
        ${MEXLIBDIR}mod3d.a \
        ${MEXLIBDIR}analysis.a \
        ${MEXLIBDIR}super3d.a \
        ${MEXLIBDIR}mod3d.a \
        ${MEXLIBDIR}analysis.a \
        ${MEXLIBDIR}mod3d.a \
        ${MEXLIBDIR}controls3d.a \
        ${MEXLIBDIR}controls.a \
        ${MEXLIBDIR}harwell.a \
        ${MEXLIBDIR}tools.a \
        ${MEXLIBDIR}daftools.a \
        ${MEXLIBDIR}mathtools.a \
        ${MEXLIBDIR}dadsblas.a \
        ${MEXLIBDIR}${LMGRLIB} \
```

```
        ${MEXLIBDIR}libparent.a \
        ${MEXLIBDIR}libcp.a \
        ${MEXLIBDIR}ctools.a \
        ${MEXLIBDIR}expressionparser.a \
        ${MEXLIBDIR}mxxdummy.o \
        ${MEXLIBDIR}tecdummy.o \
        ${MEXLIBDIR}cgdummy.o \
        ${MEXLIBDIR}ez5dummy.o

DADSLIBS= \
      ${DADSLIBDIR}blockda.o \
      ${DADSLIBDIR}revbd.o \
      ${DADSLIBDIR}xdummy.o \
      ${DADSLIBDIR}mod3d.a \
      ${DADSLIBDIR}analysis.a \
      ${DADSLIBDIR}super3d.a \
      ${DADSLIBDIR}mod3d.a \
      ${DADSLIBDIR}analysis.a \
      ${DADSLIBDIR}mod3d.a \
      ${DADSLIBDIR}controls3d.a \
      ${DADSLIBDIR}controls.a \
      ${DADSLIBDIR}harwell.a \
      ${DADSLIBDIR}tools.a \
      ${DADSLIBDIR}daftools.a \
      ${DADSLIBDIR}mathtools.a \
      ${DADSLIBDIR}dadsblas.a \
      ${DADSLIBDIR}${LMGRLIB} \
      ${DADSLIBDIR}libparent.a \
      ${DADSLIBDIR}libcp.a \
      ${DADSLIBDIR}ctools.a \
      ${DADSLIBDIR}expressionparser.a \
      ${DADSLIBDIR}mxxdummy.o \
      ${DADSLIBDIR}tecdummy.o \
      ${DADSLIBDIR}cgdummy.o \
      ${DADSLIBDIR}ez5dummy.o

FORT_OBJS = \
       Get_Obstacles2.o\
       Human_Driver.o\
       Avoid_Obstacles.o\
       Human_Steering.o\
       calcrs.o\
       calcth.o\
       checkrth.o\
       gmprd.o\
       main.o\
       Interface_Steering.o\
       fr3512.o\
       veh_steer.o\
       speed.o\
       tranxy.o

ndads3d :  ${FORT_OBJS}
      f77 -static -g -n32 -mips4 \
      -o ndads3d \
      ${FORT_OBJS} \
      ${FTNLIBS} \
      ${CFLAGS} \
```

```
      ${LDFLAGS2} \
      $(LIBS2) \
      ${DADSLIBDIR}progrm.o \
      ${DADSLIBDIR}matdummy.o \
      $(DADSLIBS) \
      -lC -lc -v

mexfile: ${FORT_OBJS}
      mex -v   \
      -output dads3d.mexsg64  $(DADSLIBDIR)dads3d.c \
      ${FORT_OBJS} \
      $(DADSLIBDIR)cxxhead.o \
      $(DADSLIBSMEX) -lC -lc

main.o:     main.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 main.f -o main.o
      @echo --- Done ---
      @echo

Avoid_Obstacles.o:   Avoid_Obstacles.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 Avoid_Obstacles.f -o
Avoid_Obstacles.o
      @echo --- Done ---
      @echo

calcrs.o:   calcrs.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 calcrs.f -o calcrs.o
      @echo --- Done ---
      @echo

calcth.o:   calcth.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 calcth.f -o calcth.o
      @echo --- Done ---
      @echo

checkrth.o:   checkrth.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 checkrth.f -o checkrth.o
      @echo --- Done ---
      @echo

Get_Obstacles2.o:   Get_Obstacles2.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 Get_Obstacles2.f -o
Get_Obstacles2.o
      @echo --- Done ---
      @echo
```

```
gmprd.o:   gmprd.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 gmprd.f -o gmprd.o
      @echo --- Done ---
      @echo

Human_Driver.o:   Human_Driver.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 Human_Driver.f -o
Human_Driver.o
      @echo --- Done ---
      @echo

Human_Steering.o:   Human_Steering.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 Human_Steering.f -o
Human_Steering.o
      @echo --- Done ---
      @echo

Interface_Steering.o:   Interface_Steering.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 Interface_Steering.f -o
Interface_Steering.o
      @echo --- Done ---
      @echo

fr3512.o:   fr3512.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 fr3512.f -o fr3512.o
      @echo --- Done ---
      @echo

veh_steer.o:   veh_steer.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 veh_steer.f -o veh_steer.o
      @echo --- Done ---
      @echo

speed.o:   speed.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 speed.f -o speed.o
      @echo --- Done ---
      @echo

tranxy.o:   tranxy.f
      @echo
      @echo Compiling $*.f...
      f77 $(CFLAGS) -static -c -g -n32 -mips4 tranxy.f -o tranxy.o
      @echo --- Done ---
      @echo
```

```
        ********************************
        * FORTRAN CODE FOR DRIVER MODEL *
        ********************************


       SUBROUTINE Avoid_Obstacles (THETRAD, DELTHRAD, NPOWER, XV, YV,
     1                             ZV, PSIRAD, JMAX, RD, TH,
     1                             XSTAR, YSTAR)
C
C
      double precision RD(200),TH(200),THETRAD, DELTHRAD,
     &                 XV,YV,ZV,PSIRAD,XSTAR,YSTAR,
     &                 PI,SUM1,SUM2,TRSTAR,TTHETA,RSTAR,
     &                 THSTAR
C
      DATA PI /3.1416/
C
C
      SUM1 = 0.0
      SUM2 = 0.0
      DO 1000 K = 1, JMAX
         L = (JMAX - K) + 1
         SUM1 = RD(L)*TH(L) + SUM1
         SUM2 = RD(L) + SUM2
C
C
1000  CONTINUE
         TRSTAR = SUM2/JMAX
         TTHETA = SUM1/SUM2
C
      CALL CALCRS(JMAX, RD, TH, THETRAD, DELTHRAD, RSTAR)
C
      CALL CALCTH(JMAX, RD, TH, THETRAD, DELTHRAD, NPOWER, THSTAR)
C
      CALL CHECKRTH(JMAX,RD,TH,RSTAR,THSTAR)

      XSTAR = RSTAR*COS(PSIRAD + THSTAR ) + XV
      YSTAR = RSTAR*SIN(PSIRAD + THSTAR ) + YV
C
C
      return
      END
C

C
C   Subroutine Get_Obstacles reads information on 3D obstacles in obstacle
C   centered coordinates, transfroms to inertial cartesian coordinates and
C   the to vehicle center coordinate both cartesian and polar.  The
subroutine
C   then calculates the distance to the obstacles along a set of sensor rays.
C
C
```

```
************************************************************************
*
************************************************************************
*


      SUBROUTINE Get_Obstacles (FIRST, TDelta, SMAX_PHI, SMAX_THETA,
     1                          RMAX,SDELTA_PHI, SDELTA_THETA,
     1                          XVE, YVE, ZVE, Psi,Theta, Phi,
     1                          SCAN_RANGE,SCAN_PHI,SCAN_THETA)
C
      dimension NPOLY(10)
C
C Surfaces in object types.  # surfaces, # corners in surface, corner indexes
      DIMENSION NUM_SURF(10), NUM_CORN(10,10), INDEX_CORN(10,10,8)
C
      dimension IOBJ_TYPE(200),ID(200)
C
C
      double precision TDelta,SMAX_PHI,SMAX_THETA,RMAX,SDELTA_PHI,
     &                 SDELTA_THETA,XVE,YVE,ZVE,Psi,Theta,Phi,
     &                 SCAN_RANGE(40,40),SCAN_THETA(40),
     &                 SCAN_PHI(40),RD(200),TH(200),XTD(18),
     &                 YTD(18),ZTD(18),XTC(10),YTC(10),ZTC(10),
     &                 XT(10,18),YT(10,18),ZT(10,18),
     &                 XCI(200),YCI(200),
     &                 ZCI(200),R_Bound(200),
     &                 E_PSI(200),E_THETA(200),E_PHI(200),
     &                 XCV(200),YCV(200),ZCV(200),RV(200),
     &                 VPHI(200),VTHETA(200),DELTA_ANGLE(200),
     &                 A(3,3),B(3,3),XTSUM,YTSUM,ZTSUM,R,PSI_E,
     &                 THETA_E,PHI_E,CXE,CYE,CZE,CX,CY,CZ,XV,YV,
     &                 ZV,PI,EPSILON
      integer num_theta,kmax,count
      LOGICAL FIRST
      CHARACTER*50 COMMENT
C
      DATA PI /3.1415926545/
      DATA EPSILON /0.000000001/
      data count /1/
C
C
      IF(FIRST) THEN
C
C    Read in obstacles from files
C
            OPEN(UNIT=9,FILE='Course_Obstacles.txt')
C
            REWIND 9

            OPEN(UNIT=4,FILE='Obstacles_Check.txt')
C
            REWIND 4
C
        READ (9,*) COMMENT
```

```
        READ (9,*), COMMENT
C
      READ (9,*) NUM_TYPE
        DO 10 I = 1, NUM_TYPE
C
C  Initialize the sum of the corner points
            XTSUM = 0.0D0
            YTSUM = 0.0D0
            ZTSUM = 0.0D0
C
            READ (9,*) COMMENT
            READ (9,*) COMMENT
C
            READ (9,*) NPOLY(I)
            READ (9,*) COMMENT
            READ (9,*) COMMENT
            DO 20 J = 1, NPOLY(I)
C
            READ (9,*), XTD(J),YTD(J),ZTD(J)
C
                XTD(J) = XTD(J)/0.3048
                YTD(J) = YTD(J)/0.3048
                ZTD(J) = ZTD(J)/0.3048
C
            PRINT 15, I,J,XTD(J),YTD(J),ZTD(J)
  15            FORMAT("TYPE = ",I5," Corner # = ",I5," XTD = ",
     1                  F10.7," YTD = ",F10.7," ZTD = ",F10.7)
                XTSUM = XTSUM + XTD(J)
                YTSUM = YTSUM + YTD(J)
                ZTSUM = ZTSUM + ZTD(J)
  20        CONTINUE
C
C  Compute the center of the object type in design coordinate
C
            XTC(I) = XTSUM/NPOLY(I)
            YTC(I) = YTSUM/NPOLY(I)
            ZTC(I) = ZTSUM/NPOLY(I)
C
            R_BOUND(I) = 0.0D0
C
            READ (9,*) COMMENT
C
            READ(9,*) NUM_SURF(I)
            DO 30 J = 1, NUM_SURF(I)
C
               READ (9,*) COMMENT
C
               READ(9,*) NUM_CORN(I,J)
C
               READ (9,*) COMMENT
C
               DO 40 K = 1, NUM_CORN(I,J)
                 READ(9,*) INDEX_CORN(I,J,K)
  40           CONTINUE
  30       CONTINUE
C
C  Shift the corner points to a coordinate system with it origin at the
```

```
C  center of volume of the the obstacle type
C
            DO 310 J = 1, NPOLY(I)
C
               XT(I,J) = XTD(J) - XTC(I)
               YT(I,J) = YTD(J) - YTC(I)
               ZT(I,J) = ZTD(J) - ZTC(I)
C
               R = SQRT(XT(I,J)**2 + YT(I,J)**2 + ZT(I,J)**2)
C
               PRINT 25, I, J, XT(I,J), YT(I,J), ZT(I,J),R
  25           FORMAT(" TYPE = ",I5,"CORNER = ",I5," XT = ",F10.7,
     1                " YT = ",F10.7," ZT = ",F10.7," R = ",F10.7)
C
               IF(R .GT.  R_BOUND(I)) THEN
                  R_BOUND(I) = R
               ENDIF
  310        CONTINUE
  10  CONTINUE
C
        READ (9,*) COMMENT
C
            READ (9,*) NUM_OBJ
C
        WRITE(4,315)
  315   FORMAT("Obj # Obj Type Corner #    ",
     1         "    CX            CY            CZ")
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
      DO 410 I = 1, NUM_OBJ
C
          READ (9,*) COMMENT
C
          READ(9,*) IOBJ_TYPE(I)
C  Input the location of the center of design coordinate system
C
           READ (9,*) COMMENT
C
         READ(9,*) XCI(I),YCI(I),ZCI(I)
C
           XCI(I) = XCI(I)/0.3048
           YCI(I) = YCI(I)/0.3048
           ZCI(I) = ZCI(I)/0.3048
C
           PRINT 605, I,XCI(I),YCI(I),ZCI(I)
  605      FORMAT("I = ",I5," XCI = ",F12.7," YCI = ",F12.7,
     1           " ZCI = ",F12.7)
C  Input the Euler angle to transform the orientation
C
           READ (9,*) COMMENT
C
           READ (9,*) PSI_E, THETA_E, PHI_E
           PRINT 650,PSI_E,THETA_E,PHI_E
  650      FORMAT(" PSI_E = ",E12.7," THETA_E = ",E12.7,
     1     " PHI_E = ",E12.7)
```

```
C
C  Convert from degrees to radians
           E_PSI(I) = PSI_E*(PI/180.0)
           E_THETA(I) = THETA_E*(PI/180.0)
           E_PHI(I) = PHI_E*(PI/180.0)
C
C
C  Checking the placement of obstacles
       K =  IOBJ_TYPE(I)
C
C  Calculate transformation matrix from object centered to inertial
coordinates
      CALL BRYANT_MATRIX(E_PSI(I), E_THETA(I), E_PHI(I), B)
C

       PRINT 755,E_PSI(I), E_THETA(I), E_PHI(I)
  755  FORMAT("E_PSI =",F12.5,"E_THETA =",F12.5,
     1            "E_PHI =",F12.5)
       PRINT 607, XCI(I),YCI(I),ZCI(I)
  607      FORMAT("XCI = ",F12.7," YCI = ",F12.7,
     1           " ZCI = ",F12.7)
CC
C
       PRINT 315

C  Transforming the corner points to vehicle Cartesian and vehicle polar
      DO 705 J1 = 1, NPOLY(K)
C
C  Reoriented corner point in object centered coordinates
C
         CALL INVERS_TRANFORMER (B,XT(K,J1),YT(K,J1),ZT(K,J1),
     1                      XCI(I),YCI(I),ZCI(I),
     1                       CX,CY,CZ)
C
C  Reoriented corner point in object centered coordinates
C
           CXE = CX
           CYE = CY
           CZE = CZ
C
           PRINT 300, I,K,J1,CXE,CYE,CZE
           WRITE(4,300)I,K,J1,CXE,CYE,CZE
  300      FORMAT(I7,2(",",I7),3(",",F12.5))
  705      CONTINUE
  410    CONTINUE
C
         ENDIF
C
      XV = XVE
      YV = YVE
      ZV = ZVE
C
C
C   Update obstacles in vehicle coordinates to account for vehicle movement
C
C   Calculate the transformation matrix to vehicle coordinates
         CALL BRYANT_MATRIX(Psi, Theta, Phi,A)
```

```
C
C   Transform object centers to vehicle Cartesian and polar coordinates
C
         CALL VEH_COORDS(A, RMAX, SMAX_PHI, SMAX_THETA, NUM_OBJ,
     1                       XCI,YCI,ZCI,XV,YV,ZV, R_BOUND,NUM_SEN,
     1                       XCV,YCV,ZCV,IOBJ_TYPE,ID,RV,VPHI,
     1                       VTHETA,DELTA_ANGLE)

C
C   Generate rays from sensor to all obstacles
C
          CALL SCAN (SMAX_PHI,SMAX_THETA,SDELTA_PHI,SDELTA_THETA,
     1               RMAX, NUM_SEN, XCV,YCV,ZCV,IOBJ_TYPE,ID,RV,VPHI,
     1               VTHETA,XCI,YCI,ZCI,NPOLY, XT,YT,ZT,NUM_SURF,
     1               NUM_CORN,INDEX_CORN,DELTA_ANGLE,R_BOUND,E_PSI,
     1               E_THETA,E_PHI,XV,YV,ZV,A,
     1               SCAN_PHI,SCAN_THETA,SCAN_RANGE)
C
       NUM_THETA = INT(SMAX_THETA/SDELTA_THETA)*2 + 1
C
C
C
       KMAX = INT(SMAX_PHI/SDELTA_PHI)*2 + 1
C
C
C
 100  FORMAT(I6)
 200  FORMAT(2E14.6)
 400  FORMAT(3E14.6)
C
C
      RETURN
      END
C


*************************************************************************
*************************************************************************



      SUBROUTINE BRYANT_MATRIX(Psi, Theta, Phi,A)
C
C    Subroutine calculates the transformation matrix between coordinate
C    system in term of three Bryant angles.  Assume a rotation about x (phi)
C    followed by a rotation about the transformed y (theta), followed by
C    rotation about the twice transformed z (psi)
C
      double precision A(3,3),Psi,Theta,Phi,COS_PHI_1,SIN_PHI_1,
     &                 COS_THETA_2,SIN_THETA_2,COS_PSI_3,
     &                 SIN_PSI_3
C
      COS_PHI_1= COS(Phi)
      SIN_PHI_1= SIN(Phi)
      COS_THETA_2= COS(Theta)
      SIN_THETA_2= SIN(Theta)
```

```
      COS_PSI_3= COS(Psi)
      SIN_PSI_3= SIN(Psi)

      A(1,1) = COS_THETA_2*COS_PSI_3
      A(1,2) = COS_THETA_2*SIN_PSI_3
      A(1,3) = -SIN_THETA_2

      A(2,1) = SIN_PHI_1*SIN_THETA_2*COS_PSI_3 - COS_PHI_1*SIN_PSI_3
      A(2,2) = SIN_PHI_1*SIN_THETA_2*SIN_PSI_3 + COS_PHI_1*COS_PSI_3
      A(2,3) = SIN_PHI_1*COS_THETA_2

      A(3,1) = COS_PHI_1*SIN_THETA_2*COS_PSI_3 + SIN_PHI_1*SIN_PSI_3
      A(3,2) = COS_PHI_1*SIN_THETA_2*SIN_PHI_3 - SIN_PHI_1*COS_PHI_3
      A(3,3) = COS_PHI_1*COS_THETA_2

      RETURN
      END
C


**************************************************************************
**************************************************************************



      SUBROUTINE TRANFORMER (A,X1,Y1,Z1,X0,Y0,Z0,X2,Y2,Z2 )
C
C   Transforms between 3D coordinate system that are rotated and displaced
C
      double precision A(3,3),X1,Y1,Z1,X0,Y0,Z0,X2,Y2,Z2
C
      X2 = A(1,1)*(X1-X0) + A(1,2)*(Y1-Y0) + A(1,3)*(Z1-Z0)
      Y2 = A(2,1)*(X1-X0) + A(2,2)*(Y1-Y0) + A(2,3)*(Z1-Z0)
      Z2 = A(3,1)*(X1-X0) + A(3,2)*(Y1-Y0) + A(3,3)*(Z1-Z0)
C
      RETURN
      END
C


**************************************************************************
**************************************************************************



      SUBROUTINE INVERS_TRANFORMER (A,X1,Y1,Z1,X0,Y0,Z0,X2,Y2,Z2 )
C
C   Inverse Transformation between coordinate systems 1 and 2 that
C   are rotated by the inverse of matrix A and displaced to point 0
C
      double precision A(3,3),X1,Y1,Z1,X0,Y0,Z0,X2,Y2,Z2
C
      X2 = (A(1,1)*X1 + A(2,1)*Y1 + A(3,1)*Z1) + X0
      Y2 = (A(1,2)*X1 + A(2,2)*Y1 + A(3,2)*Z1) + Y0
      Z2 = (A(1,3)*X1 + A(2,3)*Y1 + A(3,3)*Z1) + Z0
C
      RETURN
```

```
      END
C


*************************************************************************
*************************************************************************



      SUBROUTINE VEH_COORDS(A, RMAX, SMAX_PHI, SMAX_THETA, NUM_OBJ,
     1                      XCI,YCI,ZCI,XV,YV,ZV, R_BOUND,NUM_SEN,
     1                      XCV,YCV,ZCV,IOBJ_TYPE,ID,RV,VPHI,
     1                      VTHETA,DELTA_ANGLE)
C
C
      dimension IOBJ_TYPE(200),ID(200)
C
C
C
      double precision XCI(200),YCI(200),ZCI(200),R_Bound(200),
     &                 XCV(200),YCV(200),ZCV(200),
     &                 RV(200),VPHI(200),
     &                 VTHETA(200), DELTA_ANGLE(200),A(3,3),
     &                 XMV,YMV,ZMV,XV,YV,ZV,RANGE,EPSILON,
     &                 DELTA_A,PI,PHI,LEFT_SCAN,RIGHT_SCAN,
     &                 RATIO,THETA,UPSCAN,DOWNSCAN,RMAX,
     &                 SMAX_PHI,SMAX_THETA
C
      DATA PI /3.1415926545/
      DATA EPSILON /0.000000001/
C
      NUM_SEN = 0
        J = 0
        LAST_J = 0
C
      DO 10 I = 1, NUM_OBJ
C
            I_OBJ = IOBJ_TYPE(I)
C
            CALL TRANFORMER (A,XCI(I),YCI(I),ZCI(I)
     1                       ,XV,YV,ZV,XMV,YMV,ZMV)
C
C  Compute the range to the Jth corner of the Ith obstacle
             RANGE = SQRT(XMV**2 + YMV**2 + ZMV**2)
C  Compute the angle subtended by the bounding sphere for the Ith object
             IF(RANGE .GT.  EPSILON) THEN
                IF(RANGE .GT.  R_BOUND(I_OBJ)) THEN
                  DELTA_A = ASIN(R_BOUND(I_OBJ)/RANGE)
                ELSE
                    DELTA_A = PI/2.0
                ENDIF
             ELSE
               DELTA_A = PI/2.0
             ENDIF
C
C Check to see if the object bounding sphere is within sensor range
             IF((RANGE - R_BOUND(I_OBJ)) .LE.  RMAX) THEN
```

```
C
C           PHI = ASIN(YMV/(SQRT(XMV**2 + YMV**2)))

              IF(XMV .GE.  0.0) THEN
                PHI = ASIN(YMV/(SQRT(XMV**2 + YMV**2)))
                ELSE
                    IF(YMV .GE.  0.0) THEN
                   PHI = PI - ASIN(YMV/
     1                        (SQRT(XMV**2 + YMV**2)))
                    ELSE
                   PHI = -1.0*PI - ASIN(YMV/
     1                        (SQRT(XMV**2 + YMV**2)))
                    ENDIF
                ENDIF
C
                LEFT_SCAN = SMAX_PHI
                RIGHT_SCAN = -1.0*SMAX_PHI
C
        RATIO = (YMV/(SQRT(XMV**2 + YMV**2)))
C
C Check to see if the object bounding sphere is with the horizontal scan
                IF(((PHI - DELTA_A).LT.LEFT_SCAN) .OR.
     1            ((PHI + DELTA_A).GT.  RIGHT_SCAN)) THEN
C
C  Compute the pitch angle measure in the transformed XZ plane from Z
clockwise
C  (the XY plan is at 90 degrees)
                     IF(RANGE .GT.  EPSILON) THEN
                       THETA = ACOS(ZMV/RANGE)
                     ELSE
                        THETA = PI/2.0
                     ENDIF
C
C Calculate limit of vertical scan in spherical coordinates i.e.
C from the Z axis downwards
                    UPSCAN = (PI/2.0) - SMAX_THETA
                    DOWNSCAN = (PI/2.0) + SMAX_THETA
C
C Check to see if the object bounding sphere is with the vertical scan
                    IF(((THETA + DELTA_A).GT.  UPSCAN) .OR.
     1              ((THETA - DELTA_A).LT.DOWNSCAN)) THEN
C
C  Build an array of objects in the sensor field of view
C
                           J = J + 1
                           NUM_SEN = J
                           RV(J) = RANGE
                           VPHI(J) = PHI
                           VTHETA(J) = THETA
                           DELTA_ANGLE(J) = DELTA_A
                           XCV(J) = XMV
                           YCV(J) = YMV
                           ZCV(J) = ZMV
                           IOBJ_TYPE(J) = I_OBJ
                           ID(J) = I
                ENDIF
              ENDIF
```

```
        ENDIF
        IF(J .EQ.  LAST_J) THEN
        ENDIF
          LAST_J = J
C
  10    CONTINUE
      RETURN
      END
C
C


*************************************************************************
*************************************************************************



      SUBROUTINE SCAN (SMAX_PHI,SMAX_THETA,SDELTA_PHI,SDELTA_THETA,
     1                 RMAX, NUM_SEN, XCV,YCV,ZCV,IOBJ_TYPE,ID,RV,VPHI,
     1                 VTHETA,XCI,YCI,ZCI,NPOLY, XT,YT,ZT,NUM_SURF,
     1                 NUM_CORN,INDEX_CORN,DELTA_ANGLE,R_BOUND,E_PSI,
     1                 E_THETA,E_PHI,XV,YV,ZV,A,
     1                 SCAN_PHI,SCAN_THETA,SCAN_RANGE)

C
      dimension NPOLY(10)
C
C Surfaces in object types.  # surfaces, # corners in surface, corner indexes
      DIMENSION NUM_SURF(10), NUM_CORN(10,10), INDEX_CORN(10,10,8)
C
      dimension IOBJ_TYPE(200),ID(200)
C
C
C
      double precision SMAX_PHI,SMAX_THETA,SDELTA_PHI,RMAX,
     &                 SDELTA_THETA,XT(10,18),YT(10,18),
     &                 ZT(10,18),XCI(200),YCI(200),ZCI(200),
     &                 R_Bound(200),E_PSI(200),E_THETA(200),
     &                 E_PHI(200),XCV(200),YCV(200),
     &                 ZCV(200),RV(200),VPHI(200),VTHETA(200),
     &                 DELTA_ANGLE(200),A(3,3),B(3,3),
     &                 C(200,3,3),SCAN_RANGE(40,40),
     &                 SCAN_PHI(40),SCAN_THETA(40),PI,EPSILON,
     &                 THETA,PHI,RANGE,XV,YV,ZV
C
      DATA PI /3.1415926545/
      DATA EPSILON /0.000000001/
C
      NUM_PHI = INT(SMAX_PHI/SDELTA_PHI)*2 + 1
      NUM_THETA = INT(SMAX_THETA/SDELTA_THETA)*2 + 1
C
C  Initialize all the range along all scan ray to the sensor max
      DO 410 I2 = 1, NUM_THETA
            DO 510 J2 = 1, NUM_PHI
             SCAN_RANGE(I2,J2) = RMAX
 510        CONTINUE
 410    CONTINUE
```

```
C
C  Vertical scan
      THETA =(PI/2.0)-1.0*(SMAX_THETA + SDELTA_THETA)
      DO 100 I = 1, NUM_THETA
        THETA = THETA + SDELTA_THETA
C
C  Horizontal scan
       PHI = -1.0*(SMAX_PHI + SDELTA_PHI)
      DO 200 J = 1, NUM_PHI
           PHI = PHI + SDELTA_PHI
C
C  Check object whose bounding sphere is with the field of view
C
            DO 300 K = 1,  NUM_SEN
C
           IF((THETA .LT.  (VTHETA(K) + DELTA_ANGLE(K))).AND.
     1       (THETA .GT.  (VTHETA(K) - DELTA_ANGLE(K))) .AND.
     1           (PHI .LT.  (VPHI(K) + DELTA_ANGLE(K))).AND.
     1       (PHI .GT.  (VPHI(K) - DELTA_ANGLE(K)))) THEN
C
C   Ray hits bounding sphere
            I_OBJ = IOBJ_TYPE(K)
            ID2 = ID(K)
C
            CALL SCAN_SURF(RMAX, PHI, THETA, XCI(ID2), YCI(ID2),
     1                     ZCV(ID2), I_OBJ,XV,YV,ZV,E_PSI(ID2),
     1                     E_THETA(ID2),E_PHI(ID2), NUM_SURF,
     1                     NUM_CORN,INDEX_CORN,NPOLY(I_OBJ),
     1                     XT,YT,ZT,A,RANGE)
C
              IF(RANGE .LT.  SCAN_RANGE(I,J)) THEN
C
                 SCAN_THETA(I) = THETA
                 SCAN_PHI(J) = PHI
                 SCAN_RANGE(I,J) = Range
C
               ENDIF
C
           ELSE
C
C       The Kth Scan ray miss all objects.  Set range for the ray to max
sensor range
              SCAN_THETA(I) = THETA
              SCAN_PHI(J) = PHI
C
           ENDIF
 300       CONTINUE
 200     CONTINUE
 100  CONTINUE
C
      RETURN
      END
C
C
      SUBROUTINE SCAN_SURF(RMAX,SCAN_PHI,SCAN_THETA,XC,YC,
     1                     ZC, I, XV,YV,ZV,
     1                     PSI_E,THETA_E,PHI_E,
```

```
     1                    NUM_SURF,NUM_CORN,INDEX_CORN,
     1                    N_POLY,XT,YT,ZT,A,RANGE)
C
C
C
C  Surfaces in object types.  # surfaces, # corners in surface, corner
indexes
      DIMENSION NUM_SURF(10), NUM_CORN(10,10), INDEX_CORN(10,10,8)
C
C
      double precision XT(10,18),YT(10,18),ZT(10,18),XCV(18),
     &                 YCV(18),ZCV(18),A(3,3),B(3,3),E_PSI(200),
     &                 E_THETA(200),E_PHI(200),PI,EPSILON,RMAX,
     &                 SCAN_PHI,SCAN_THETA,XC,YC,ZC,XV,YV,ZV,
     &                 PSI_E,THETA_E,PHI_E,RANGE,XCR,YCR,ZCR,
     &                 CX,CY,CZ,PHI_MIN,PHI_MAX,THETA_MIN,
     &                 THETA_MAX,PHI,THETA,SURF_RANGE
C
      DATA PI /3.1415926545/
      DATA EPSILON /0.000000001/

C
C  Calculate transformation matrix from object centered to inertial
coordinates
      CALL BRYANT_MATRIX(PSI_E, THETA_E, PHI_E, B)

C
C  Transforming the corner points to vehicle Cartesian and vehicle polar
      DO 100 J1 = 1,N_POLY
C
C  Reoriented corner point in object centered coordinates
C
C
         CALL TRANFORMER (B,XT(I,J1),YT(I,J1),ZT(I,J1),
     1                    0.0D0,0.0D0,0.0D0,
     1                     XCR,YCR,ZCR)
C
C Translate corner point to new location in initerial coordinates system
           CX = XCR + XC
           CY = YCR + YC
           CZ = ZCR + ZC
C
C  Transforming the corner points from initial to vehicle Cartesian
         CALL TRANFORMER (A,CX,CY,CZ,
     1                     XV,YV,ZV,
     1                     XCV(J1),YCV(J1),ZCV(J1))
C
 100     CONTINUE
C  Checking each surface to see if the ray passes through it
      RANGE = RMAX
C
      DO 200 J = 1, NUM_SURF(I)
C
           PHI_MIN = PI
           PHI_MAX = -1.0*PI
           THETA_MIN = PI
           THETA_MAX = 0.0
```

```
C
           DO 300 K = 1, NUM_CORN(I,J)
C Compute the range to the Kth corner of the Jth surface of the Ith object
                L =  INDEX_CORN(I,J,K)
C
C  Compute the yaw angle measure in the XY plan counter clockwise from X
              IF(XCV(L) .GE.  0.0) THEN
                PHI = ASIN(YCV(L)/(SQRT(XCV(L)**2 + YCV(L)**2)))
                ELSE
                    IF(YCV(L) .GE.  0.0) THEN
                    PHI = PI - ASIN(YCV(L)/(SQRT(XCV(L)**2
     1                    + YCV(L)**2)))
                    ELSE
                    PHI = -1.0*(PI) - ASIN(YCV(L)/
     1                         (SQRT(XCV(L)**2 + YCV(L)**2)))
                    ENDIF
                ENDIF
C
C Check for max and min horizontal angle
                IF(PHI .LT.  PHI_MIN) THEN
                    PHI_MIN = PHI
                ENDIF
                 IF(PHI .GT.  PHI_MAX) THEN
                    PHI_MAX = PHI
                ENDIF
C
C  Compute the pitch angle measure in the transformed XZ plane from Z
clockwise
C  (the XY plan is at 90 degrees)
C
                THETA =(PI/2)- ASIN(ZCV(L)/(SQRT(XCV(L)**2 + YCV(L)**2 +
     1                       ZCV(L)**2)))
C
C  Check for max and min horizontal angle
                IF(THETA .LT.  THETA_MIN) THEN
                    THETA_MIN = THETA
                ENDIF
C
                 IF(THETA .GT.  THETA_MAX) THEN
                    THETA_MAX = THETA
                ENDIF
C
 300        CONTINUE
C
            IF ((SCAN_PHI .LT.  PHI_MAX) .AND.  (SCAN_PHI.  GT.  PHI_MIN)
     1      .AND.  (SCAN_THETA .LT.  THETA_MAX) .AND.
     1      (SCAN_THETA.  GT.  THETA_MIN)) THEN
C
C  Compute the intersection of the scan ray with the surface
C
                CALL RAY_INTERSECT(RMAX, SCAN_THETA, SCAN_PHI, I, J,
     1                             NUM_CORN, INDEX_CORN, XCV, YCV, ZCV,
     1                             SURF_RANGE)
C
                IF (SURF_RANGE .LT.  RANGE) THEN
                      RANGE = SURF_RANGE
                ENDIF
```

```
C
            ENDIF
 200  CONTINUE
C
      RETURN
      END
C


*************************************************************************
*************************************************************************



      SUBROUTINE RAY_INTERSECT(RMAX, SCAN_THETA, SCAN_PHI, I, J,
     1                         NUM_CORN, INDEX_CORN, XCV, YCV, ZCV,
     1                         RANGE)
C
C  Surfaces in object types.  # surfaces, # corners in surface, corner
indexes
      DIMENSION NUM_SURF(10), NUM_CORN(10,10), INDEX_CORN(10,10,8)
C
C
      double precision XCV(18),YCV(18),ZCV(18),EPSILON,RMAX,
     &                 SCAN_THETA,SCAN_PHI,RANGE,A1,A2,A3,B1,
     &                 B2,B3,C1,C2,C3,DV,A,B,C,DP,XUSR,YUSR,
     &                 ZUSR,COS_PSI,DSR,XSR,YSR,ZSR,XSUM,YSUM,
     &                 ZSUM,XSC,YSC,ZSC,AA,AB,AC,U0,V0,U1,V1,
     &                 U2,V2,D0,D1,D2,ALPHA,BETA,ABSUM
      LOGICAL FOUND

      DATA EPSILON /0.000000001/
C
      K0 = INDEX_CORN(I,J,1)
      K1 = INDEX_CORN(I,J,2)
      K3 = INDEX_CORN(I,J,NUM_CORN(I,J))
C
C Define the vector that make up the edges of the surface
      A1 = XCV(K1) - XCV(K0)
      A2 = YCV(K1) - YCV(K0)
      A3 = ZCV(K1) - ZCV(K0)
C
      B1 = XCV(K3) - XCV(K0)
      B2 = YCV(K3) - YCV(K0)
      B3 = ZCV(K3) - ZCV(K0)
C
C Defining the vector cross product.  Vector will be normal to the surface
      C1 = A2*B3 - A3*B2
      C2 = A3*B1 - A1*B3
      C3 = A1*B2 - A2*B1
C
C Calculate the length of the vector

      DV = SQRT(C1**2.0 + C2**2.0 + C3**2.0)
C
C  Elements of the unit normal vector to the plan of the surface
      A = C1/DV
```

```
      B = C2/DV
      C = C3/DV
C  Calculating the distance from the coordinate center (vehicle center) to
the
C  plane containing  the surface.
      DP = ABS(A*XCV(K1) + B*YCV(K1) + C*ZCV(K1))
C
C  Calculating the elements of a unit vector parallel with the scan ray
      XUSR = SIN(SCAN_THETA)*COS(SCAN_PHI)
      YUSR = SIN(SCAN_THETA)*SIN(SCAN_PHI)
      ZUSR = COS(SCAN_THETA)
C
C  For unit vectors starting at the origin the direction cosines are equal
C  to the elements of the vector.  Using the direction cosines, the solid
C  angle between the surface normal and the scan ray is the following
      COS_PSI = ABS((A*XUSR + B*YUSR + C*ZUSR))
C
C  Distance from the vehicle to the plane containing the surface along
C  the scan ray
      IF(COS_PSI .GT.  EPSILON) THEN
        DSR = DP/COS_PSI
      ELSE
        DSR = RMAX
      ENDIF
C
C  Point at which the scan ray intersect the plane containing the surface
      XSR = DSR*XUSR
      YSR = DSR*YUSR
      ZSR = DSR*ZUSR
C
C  Finding the center of the surface
      XSUM = 0.0
      YSUM = 0.0
      ZSUM = 0.0
      DO 100 K = 1, NUM_CORN(I,J)
        L = INDEX_CORN(I,J,K)
        XSUM = XCV(L) + XSUM
        YSUM = YCV(L) + YSUM
        ZSUM = ZCV(L) + ZSUM
 100  CONTINUE
      XSC = XSUM/NUM_CORN(I,J)
      YSC = YSUM/NUM_CORN(I,J)
      ZSC = ZSUM/NUM_CORN(I,J)
C
C  Divide the surface in triangles.  Each triangle contains two corner points
C  and the center point.  Check each triangle to see if the point where the
C  scan ray contacts the plane containing the surface is inside.
C
C  Check each triangle
C
       FOUND = .FALSE.
      DO 200 K = 1, NUM_CORN(I,J)
        L = INDEX_CORN(I,J,K)
C
        KP1 = K + 1
        IF(KP1 .GT.  NUM_CORN(I,J)) THEN
           KP1 = 1
```

```
        ENDIF
        M = INDEX_CORN(I,J,KP1)
C
      AA = ABS(A)
        AB = ABS(B)
        AC = ABS(C)
C
        IF ((AC .GE.  AA) .AND.  (AC .GE.  AB)) THEN
           U0 = XSR - XSC
           V0 = YSR - YSC
           U1 = XCV(L) - XSC
           V1 = YCV(L) - YSC
           U2 = XCV(M) - XSC
           V2 = YCV(M) - YSC
C
         ELSE IF((AB .GE.  AA) .AND.  (AB .GE.  AC)) THEN
           U0 = XSR - XSC
           V0 = ZSR - ZSC
           U1 = XCV(L) - XSC
           V1 = ZCV(L) - ZSC
           U2 = XCV(M) - XSC
           V2 = ZCV(M) - ZSC
C
        ELSE
           U0 = YSR - YSC
           V0 = ZSR - ZSC
           U1 = YCV(L) - YSC
           V1 = ZCV(L) - ZSC
           U2 = YCV(M) - YSC
           V2 = ZCV(M) - ZSC
C
        ENDIF
C
        D0 = U1*V2 - U2*V1
        D1 = U0*V2 - U2*V0
        D2 = U1*V0 - U0*V1
C
        ALPHA = D1/D0
        BETA = D2/D0
        ABSUM = ALPHA + BETA
C
        IF((ALPHA .GE.  0.0) .AND.  (BETA .GE.  0.0) .AND.
     1    (ABSUM .LE.  1.0)) THEN
C
            FOUND = .TRUE.
C
        ENDIF
C
 200  CONTINUE
C
      IF(FOUND) THEN
         RANGE = DSR
      ELSE
         RANGE = RMAX
      ENDIF
C
C
```

```
      RETURN
      END


************************************************************************
************************************************************************

C
      SUBROUTINE Human_Driver (NPOWER,RMAX,SMAX_PHI,
     1                         SMAX_THETA,SDELTA_PHI,
     &                         SDELTA_THETA,
     &                         X_Sen, Y_Sen, Z_Sen,
     &                         Psi_Sen,Theta_Sen,Phi_Sen)
C
      double precision RMAX,SMAX_PHI,SMAX_THETA,SDELTA_PHI,
     &                 SDELTA_THETA,GRAV,TICYCL,TSS,DMAX,XP,
     &                 YP,TAUMEM,TFF,RM,A,B,RI,PSIO,TLAST,
     &                 DFWLST,TILAST,DMEM,XT,YT,
     &                 CAF,CAR,WHBS,WF,WR,U,X_Sen,
     &                 Y_Sen, Z_Sen,Psi_Sen,Theta_Sen,Phi_Sen
C
C---COMMON blocks-----------------------------------------------------
C
      COMMON /DRVST1/ GRAV, TICYCL, TSS, DMAX, XP(100), YP(100), TAUMEM,
     1       TFF, RM, A, B, RI, PSIO, NTF, NP, TLAST, DFWLST, TILAST,
     2       DMEM(1000,2), XT(100), YT(100)
      SAVE/DRVST1/
      COMMON /DRIV/ CAF, CAR, WHBS, WF, WR, U
      SAVE/DRIV/
      COMMON /INOUT/ R, W
      SAVE/INOUT/
C
C---COMMON Variables--------------------------------------------------
C  R.....Driver Model Input I/O unit ("DMINPUT.INP")
C  W.....Driver Model Output I/O unit ("DMOUTPUT.OUT")
C
C---DRIV.BLK common block variables
C
C  CAF...total cornering stiffness of tires on left front susp (lb/rad)
C  CAR...total cornering stiffness of tires on left rear susp (lb/rad)
C  WHBS..wheelbase of vehicle (center-line of front & rear susp) (ft)
C  WF....static load on front suspension (lb)
C  WR....static load on rear suspension (lb)
C  U.....initial velocity (ft/sec)
C
C---DRVST1.BLK common block variables---------------------------------
C23456789012345678901234567890123456789012345678901234567890123456789012
C  GRAV.....gravitational constant
C  TICYCL...driver model sample time (sec)
C  TSS......minimum preview time (sec)
C  DMAX.....upper bound on front wheel angle steer (rad)
C  XP, YP...x-y path coords(SAE) wrt inertial coords (input) (ft)
C  TAUMEM...driver transport time dealy (input parameter) (sec)
C  TFF......driver model preview time (input parameter) (sec)
C  RM.......vehicle mass (slug)
C  A........distance from c.g.  to front suspension center-line (ft)
C  B........distance from c.g.  to rear suspension center-line (ft)
```

```
C  RI.......total vehicle yaw inertia (slug-ft)
C  PSIO.....current yaw angle reference value (rad)
C  NTF......number of points in the preview time interval
C  NP.......number of points in the x-y trajectory table
C  TLAST....last time driver model calculated a steer value (sec)
C  DFWLAST..last value of steer calculated by driver model (rad)
C  TILAST...last sample time driver model calculated a steer value (sec)
C  DMEM.....2-dim array (time & steer history) used in delay calculat'n
C  XT, YT...transformation of XP,YP in vehicle body axes (ft)
C
C---Local variables-------------------------------------------------------
C
      CHARACTER*80 COMMENT
C
C  Set Pi
      DATA PI /3.1416/
C  Set Acceleration of gravity (ft/sec**2)
      DATA GRAV /32.16666/
C  Set driver model sample time
C      DATA TICYCL /0.000001/
      data ticycl /1.0D-6/
C  Set minimum driver model preview time
      DATA TSS /0.0D0/
C  Set upper bound on front wheel steering angle
C
C
      DMAX = 0.2
C
C
      OPEN(UNIT=3,FILE='Human_Driver.txt')
C
      REWIND 3
C
C
C
      READ(3,*) COMMENT
C
      READ(3,*) RMAX,ACCR, SMAX_PHI, SMAX_THETA
C
C
C Convert to ft
      RMAX = RMAX/0.3048
C
C  Convert to radians
      SMAX_PHI = SMAX_PHI*(PI/180.)
      SMAX_THETA = SMAX_THETA*(PI/180.)
C
      READ(3,*) COMMENT
C
      READ(3,*) SDELTA_PHI, SDELTA_THETA

      SDELTA_PHI = SDELTA_PHI*(PI/180.)
      SDELTA_THETA = SDELTA_THETA*(PI/180.)
C
      READ(3,*) COMMENT
C
      READ(3,*) NPOWER
```

```
C
C
      READ(3,*) COMMENT
C
      READ(3,*) X_Sen, Y_Sen, Z_Sen
C
      READ(3,*) COMMENT
C
      READ(3,*) Psi_Sen,Theta_Sen,Phi_Sen
C
      Psi_Sen = Psi_Sen*(PI/180.)
      Theta_Sen = Theta_Sen*(PI/180.)
      Phi_Sen = Phi_Sen*(PI/180.)
C
      DO 90 I = 1, 1000
             DMEM(I,1) = 0.
             DMEM(I,2) = -1.
  90  CONTINUE
C
      RETURN
      END
C

C


**********************************************************************
**********************************************************************



      SUBROUTINE HUMAN_STEERING(T,Tdelta,Y,DFW,DFWNOW,
     &                          XSTAR,YSTAR)
C
      double precision T,Y(5),DFW,DFWNOW,XSTAR,YSTAR,GRAV,TICYCL,
     &                 TSS,DMAX,XP,YP,TAUMEM,TFF,RM,A,B,RI,PSI0,
     &                 TLAST,DFWLST,TILAST,DMEM,XT,YT,AAA,BBB,
     &                 CCC,DDD,RATIO,Tdelta
C
      COMMON /DRVST1/ GRAV,TICYCL,TSS,DMAX,XP(100),YP(100),TAUMEM,
     1     TFF,RM,A,B,RI,PSI0,NTF,NP,TLAST,DFWLST,TILAST,
     2     DMEM(1000,2), XT(100), YT(100)
      SAVE/DRVST1/
C
C
      COMMON /VEHTYP/ AAA, BBB, CCC, DDD, RATIO
      SAVE/VEHTYP/
C
      LOGICAL FIRST
      CHARACTER*80 COMMENT
C
      DATA FIRST /.TRUE./
C
      IF(FIRST) THEN
         OPEN(UNIT=5,FILE='Human_Steering.txt')
C
         REWIND 5
```

```
C
        READ(5,*) COMMENT
C
        READ(5,*) TAUMEM
C
        READ(5,*) COMMENT
C
        READ(5,*) TFF
C
        READ(5,*) COMMENT
C
        READ(5,*) TSS
C
        READ(5,*) COMMENT
C
        READ(5,*) DFWNOW
C
        READ(5,*) COMMENT
C
        READ(5,*) AAA
C
        READ(5,*) COMMENT
C
        READ(5,*) BBB
C
        READ(5,*) COMMENT
C
        READ(5,*) CCC
C
        READ(5,*) COMMENT
C
        READ(5,*) DDD
C
        READ(5,*) COMMENT
C
        READ(5,*) RATIO
C
        PRINT 75, TAUMEM, TFF
 75     FORMAT (' ', /, ' ', T20, 'DRIVER TRANSPORT LAG (SEC) :', T60,
     1      F4.2, /, ' ', T20, 'END OF PREVIEW INTERVAL (SEC) :', T60,
     2      F4.2/)
C
        NTF = 10
        TLAST = 0.0D0
        DFWLAST = 0.0D0
        TILAST = 0.0D0
        DFW = 0.0D0
        FIRST = .FALSE.
      ENDIF
C
      DO 10 I = 1, 5
C
        write(38,*) "I = ",I," Y = ",Y(I)
C
  10  CONTINUE
C
C
```

```
      CALL NEWDRIVER(T, Y, DFW, DFWNOW,XSTAR,YSTAR)
C
      DFWNOW = DFW
      RETURN
      END


************************************************************************
************************************************************************
C
C***********************************************************************
C
C   ***  Closed-Loop Steer Calculation Subroutine  ***
C
C  NEWDRIVER: Computes closed-loop steering control during the simulation.
C             NEWDRIVER is a modification of the original DRIVER
C             subroutine described below.
C
C

C-----------Author and Modification Section----------------------------
C
C  Author:   C.  C.  MacAdam
C
C  Date written:  01/01/88
C
C  Written on:
C
C  Modifications:
C
C-----------------------------------------------------------------------
C
C-----------Algorithm Description--------------------------------------
C
C  Purpose and use:
C
C  Error conditions:
C
C  References:
C
C       (1)  MacAdam, C.  C.  "Development of Driver/Vehicle Steering
C               Interaction Models for Dynamic Analysis,"  Final
C               Technical Report, U.S.  Army Tank Automotive Command
C               Contract No.  DAAE07-85-C-R069, The University of
C               Michigan Transportation Research Inst.  December 1, 1988
C
C       (2)  MacAdam, C.  C.  "Application of an Optimal Preview Control
C               for Simulation of Closed-Loop Automobile Driving,"
C               IEEE Transactions on Systems, Man, and Cybernetics,
C               Vol.  11, June 1981.
C
C       (3)  MacAdam, C.  C.  "An Optimal Preview Control for Linear
C               Systems," Journal of Dynamic Systems, Measurement,
C               and Control, ASME, Vol.  102, No.  3, September 1980.
C
C
C  Machine dependencies:  none
```

```
C
C  Called by:  STEERING.f
C
C-----------------------------------------------------------------------
C
      SUBROUTINE NEWDRIVER(X, Y, DFW, DFWNOW,XSTAR,YSTAR)
C
C----------------Variable Descriptions---------------------------------
C
C---Arguments passed:
C
C  -> X......time in the simulation (sec)
C  -> Y......current driver model state vector obtained from DADS.
C         Driver model state vector of dimension 5 comprised of the
C         following physical quantities: (1) inertial lateral
C         displacement (ft), (2) lateral veloc in body frame (ft/s),
C         (3) yaw rate global (rad/s), (4) SAE global yaw angle (rad),
C         (5) global forward displacement (ft).
C
C  <- DFW....closed-loop steering control returned to DADS (returned)
C  -> DFWNOW.current steering angle (average) of front wheels, passed
C         in after effects of roll-steer, compliance, etc.
C XSTAR - Previewed or desired longitudinal position
C YSTAR - Previewed or desired lateral position
C
C -----------------------------------------------------------------------
      INTEGER R, W
C
      double precision Y(5),YC(5),DUMV11(4),DUMV1(4),VECM(4),
     &                 DUMM1(4,4),DUMM2(4,4),X,DFW,DFWNOW,
     &                 XSTAR,YSTAR,GRAV,TICYCL,TSS,DMAX,
     &                 XP,YP,TAUMEM,TFF,RM,A,B,RI,PSI0,TLAST,
     &                 DFWLST,TILAST,DMEM,XT,YT,CAF,CAR,WHBS,
     &                 WF,WR,U,TTT,TTT1,G,AAA,BBB,CCC,
     &                 DDD,RATIO,CCAF1,CCAF2,CCAR1,CCAR2,
     &                 FFZL1,FFZL2,FFZL3,FFZL4,DMVELC,PI,
     &                 ANGLE,DELTA_ANGLE,XV,YV,T,EPSI,Y0,X0,
     &                 EPSY2,TSUM,SSUM,CAFTEM,CARTEM,TJI,XCAR,
     &                 YPATH,S1,EP,T1,TTAB,DFWLAST
C
C---COMMON blocks-------------------------------------------------------
C
      COMMON /DRVST1/ GRAV, TICYCL, TSS, DMAX, XP(100), YP(100), TAUMEM,
     1     TFF, RM, A, B, RI, PSI0, NTF, NP, TLAST, DFWLST, TILAST,
     2     DMEM(1000,2), XT(100), YT(100)
      SAVE/DRVST1/
C
      COMMON /DRIV/ CAF, CAR, WHBS, WF, WR, U
      SAVE/DRIV/
C
      COMMON /INOUT/ R, W
      SAVE/INOUT/
C
      COMMON /TRSSTR/ TTT(4,4,10), TTT1(4,4,10), G(4)
      SAVE /TRSSTR/
C
      COMMON /VEHTYP/ AAA, BBB, CCC, DDD, RATIO
```

```
      SAVE/VEHTYP/
C
C  Get Tire Cornering Stiffnesses, Vertical Tire Loads, and Speed
C  from DADS Through Common Block DMTIR
C
      COMMON/DMTIR/CCAF1,CCAF2,CCAR1,CCAR2,FFZL1,FFZL2,FFZL3,FFZL4,
     +    DMVELC
C
C
C---COMMON Variables--------------------------------------------------
C  R.....Driver Model Input I/O unit ("DMINPUT.INP")
C  W.....Driver Model Output I/O unit ("DMOUTPUT.OUT")
C
C---DRIV.BLK common block variables----------------------------------
C
C  Initial Values from Time Zero:
C
C  CAF...total cornering stiffness of tires on left front susp (lb/rad)
C  CAR...total cornering stiffness of tires on left rear susp (lb/rad)
C  WHBS..wheelbase of vehicle (center-line of front & rear susp) (ft)
C  WF....static load on front suspension (lb)
C  WR....static load on rear suspension (lb)
C  U.....initial velocity (ft/sec)
C
C---DMTIR.BLK common block variables----------------------------------
C
C  Updates during simulation run:
C
C  CCAF1...Left front tire cornering stiffness from DADS during run
C  CCAF2...Right front tire cornering stiffness from DADS during run
C  CCAR1...Left rear tire cornering stiffness from DADS during run
C  CCAR2...Right rear tire cornering stiffness from DADS during run
C  FFZL1...Left front tire vertical load from DADS during run
C  FFZL2...Right front tire vertical load from DADS during run
C  FFZL3...Left rear tire vertical load from DADS during run
C  FFZL4...Right rear tire vertical load from DADS during run
C  DMVELC..Forward speed from DADS
C
C---DRVST1.BLK common block variables---------------------------------
C
C  GRAV....gravitational constant
C  TICYCL..driver model sample time (sec)
C  TSS.....minimum preview time (sec)
C  DMAX....upper bound on front wheel angle steer (rad)
C  XP,YP...x-y path coords(SAE) wrt inertial coords (input) (ft)
C  TAUMEM..driver transport time dealy (input parameter) (sec)
C  TFF.....driver model preview time (input parameter) (sec)
C  RM......vehicle mass (slug)
C  A.......distance from c.g.  to front suspension center-line (ft)
C  B.......distance from c.g.  to rear suspension center-line (ft)
C  RI......total vehicle yaw inertia (slug-ft)
C  PSIO....current yaw angle reference value (rad)
C  NTF.....number of points in the preview time interval
C  NP......number of points in the x-y trajectory table
C  TLAST...last time driver model calculated a steer value (sec)
C  DFWLST..last value of steer calculated by driver model (rad)
C  TILAST..last sample time driver model calculated a steer value (sec)
```

```
C  DMEM....2-dim array (time & steer history) used in a delay calculat'n
C  XT,YT...transformation of XP,YP in vehicle body axes (ft)
C
C---TRSSTR.BLK common block variables-----------------------------------
C
C  TTT.....transition matrix at 10 discrete points in preview interval
C  TTT1....integral of trans matrix wrt preview time
C  G.......vector of control gain coefficients
C
C---Local variables-----------------------------------------------------
C
C  YC......local (body-axis based) copy of state vector Y
C  VECM....observer vector - lateral displacement from state vector
C  DUMV1...work vector
C  DUMV11..work vector
C  DUMM1...work matrix
C  DUMM2...work matrix
C  T.......time in the simulation (sec)
C  EPSI....yaw angle between body axis and current index value, PSIO
C  PSIO....current nominal value of yaw angle used for linearization
C  NP......number of points in x-y path table
C  XP,YP...x-y inertial path table (input) (ft)
C  XT,YT...x-y path table transformed to body axis (PSIO) system (ft)
C  EPSY2...cumulative preview path error squared
C  EPSI....mean squared value of cumulative preview path error
C  TSUM....scalar work quantity
C  SSUM....scalar work quantity
C  DFWLAST.steering control from last calculation (rad)
C  TJI.....preview time ahead from present time value (sec)
C  I,J,K...integer counters
C  XCAR....preview distance ahead in feet (ft)
C  XO......present forward position of vehicle c.g.  (ft)
C  TTAB....current time less the driver delay, TAUMEM.  Used to access
C            the delayed driver response stored in DMEM array.  (sec)
C  S1......scalar work quantity
C  T1......scalar work quantity
C  EP......previewed path error (ft)
C
C---Functions and Subroutines-------------------------------------------
C
      EXTERNAL  GMPRD
C
C-----------------------------------------------------------------------
C
C---------Process Block-------------------------------------------------
C
C
      DATA VECM /1.0D0,0.0D0,0.0D0,0.0D0/
      DATA PI /3.141592/
      DATA ANGLE/1.57079/
      DATA DELTA_ANGLE/0.00628319/
C
C
      NP=1
      XP(1) = XSTAR - XV
      YP(1)=YSTAR - YV
C
```

```
      T = X
C
      EPSI = ABS(Y(4) - PSIO)
C
      DO 10 I = 1, 5
C
  10  YC(I) = Y(I)
C
C  Update Coordinate Transformation
C
      PSIO = Y(4)
      DO 20 J = 1, NP
           XT(J) =  XP(J) * COS(PSIO) + YP(J) * SIN(PSIO)
  20       YT(J) = -XP(J) * SIN(PSIO) + YP(J) * COS(PSIO)
C
  30  Y0 = -Y(5) * SIN(PSIO) + Y(1) * COS(PSIO)
      X0 =  Y(5) * COS(PSIO) + Y(1) * SIN(PSIO)
      YC(1) = Y0
      YC(4) = Y(4) - PSIO
      EPSY2 = 0.0D0
      TSUM = 0.0D0
      SSUM = 0.0D0
C
C SET DFW EQUAL TO THE PREVIOUS STEERING COMMAND
C
      DFW = DFWLST
C
C  Return if time from last calculation less than sample interval
C
      IF (T - TILAST .LE.  TICYCL) RETURN
C      if(t-tilast.le.ticycl)then
C         return
C      endif
C
C  THE NEXT 6 LINES OF EXECUTABLE CODE MAY BE COMMENTED OUT TO BYPASS
C     CONTINUOUS UPDATING OF THE TRANSITION MATRICES, IF NOT REQUIRED.
C     SEE SECTION 5.8 OF REFERENCE 1.
C
C  UPDATE TIRE CORNERING STIFFNESSES AND VEHICLE VELOCITY
C     AND RECALCULATE TRANSITION MATRIX:
C
C
      CAFTEM = (CCAF1*FFZL1+CCAF2*FFZL2) / (FFZL1+FFZL2)
      CARTEM = (CCAR1*FFZL3+CCAR2*FFZL4) / (FFZL3+FFZL4)
      CAF = CAFTEM
      CAR = CARTEM
C
C
C  UPDATE TRANSITION MATRICES
C
C
      CALL TRANS
C
C
C  LOOP TO CALCULATE OPTIMAL PREVIEW CONTROL PER REFERENCES 2 & 3:
C     (NTF POINTS WITHIN THE PREVIEW INTERVAL)
C
```

```
      DO 50 I = 1, NTF
           TJI = (TFF - TSS) / NTF * I + TSS
           DO 40 J = 1, 4
                DO 40 K = 1, 4
                     DUMM1(J,K) = TTT1(J,K,I)
                     DUMM2(J,K) = TTT(J,K,I)
  40       CONTINUE
C
           CALL GMPRD(VECM, DUMM1, DUMV11, 1, 4, 4)
           CALL GMPRD(VECM, DUMM2, DUMV1, 1, 4, 4)
           CALL GMPRD(DUMV1, YC, T1, 1, 4, 1)
C
C  Get observed path input, YPATH, within preview interval at XCAR ft:
C
      XCAR = X0 + U * TJI
C
      YPATH=YT(1)
C
      CALL GMPRD(DUMV11, G, S1, 1, 4, 1)
C
C  EP is the previewed path error at this preview point
C
      EP = T1 + S1 * DFWNOW - YPATH
C
C
      TSUM = TSUM + EP * S1
      SSUM = SSUM + S1 * S1
C
C  Cumulative preview error calculation (unrelated to control)
C
      EPSY2 = EPSY2 + EP * EP * (TFF - TSS) / NTF
C
  50  CONTINUE
C
C  Cumulative preview error calculation ( unrelated to control)
C
      EPSY2 = SQRT(EPSY2) / (TFF - TSS)
C
C  Optimal value - no delay yet.
C
      DFW = -TSUM / SSUM + DFWNOW
C
C
C  Maximum steer bound set at DMAX (arbitrary)
C
       If (ABS(DFW) .GT.  DMAX) THEN
          IF (DFW .LT.  0.0) THEN
             DFW = -1.0*DMAX
          ELSE
             DFW = DMAX
          ENDIF
       ENDIF
C
C
C  Store steer history and corresponding times in DMEM.
C  Retrieve steer delayed by TAUMEM sec and return as
C      delayed driver steer control, DFW.
```

```
C
      DO 60 J = 1, 2
           DO 60 I = 1, 999
  60  DMEM(1001 - I,J) = DMEM(1000 - I,J)
      DMEM(1,1) = DFW
      DMEM(1,2) = T
      TTAB = T - TAUMEM
C
      DO 70 I = 1, 999
C
           IF (DMEM(I + 1,2) .LE.  TTAB .AND.  DMEM(I,2) .GE.  TTAB)
     1         GO TO 90
  70  CONTINUE
      WRITE (W,80)TAUMEM,DFW,X
  80  FORMAT ('0', '***** TAUMEM PROBABLY TOO LARGE *****',
     &             /,3(1X,G12.6))
      CALL EXIT
  90  DFW = DMEM(I,1)
C
C   If simulation time is less than human reaction time steering
C   angle set to zero
       IF(TTAB .LT.  0.0) THEN
          DFW = 0.0D0
       ENDIF
C
C  Save steer and time values for next calculation.
C
      DFWLST = DFW
      TLAST = X
      TILAST = X
C
      RETURN
      END


**********************************************************************
**********************************************************************



C
C**********************************************************************
C**********************************************************************
C
C  ***  Transition Matrix Calculation Subroutine  ***
C
C
C TRANS: Computes transition matrix (and integral) of the linearized
C        system, F, described in references.  Result stored in common
C        arrays TTT and TTT1 respectively.  10 pts per preview interval.
C
C-----------Author and Modification Section----------------------------
C
C  Author:  C.  C.  MacAdam
C
C  Date written:  01/01/88
C
```

```
C  Written on:
C
C  Modifications:
C
C-------------------------------------------------------------------
C-------------Algorithm Description-----------------------------------
C
C  Purpose and use: Used by the driver model in predicting future states
C
C  Error conditions:
C
C  Machine dependencies: none
C
C  Called by:  DRIVGO
C
C-------------------------------------------------------------------
C
      SUBROUTINE TRANS
C
C------------Variable Descriptions-----------------------------------
C
C---Arguments passed: None
C
      INTEGER R, W
      double precision SV(4),SD(4),SVI(4),GRAV,TICYCL,TSS,DMAX,XP,YP,
     &                 TAUMEM,TFF,RM,A,B,RI,PSIO,TLAST,DFWLAST,TILAST,
     &                 DMEM,XT,YT,CAF,CAR,WHBS,WF,WR,U,TTT,TTT1,G,
     &                 AAA,BBB,CCC,DDD,RATIO,DELT,A1,B1,A2,B2,C1,C2,
     &                 ULAST,TIME
C
C---COMMON blocks---------------------------------------------------
C
      COMMON /DRVST1/ GRAV, TICYCL, TSS, DMAX, XP(100), YP(100), TAUMEM,
     1       TFF, RM, A, B, RI, PSIO, NTF, NP, TLAST, DFWLAST, TILAST,
     2       DMEM(1000,2), XT(100), YT(100)
      SAVE/DRVST1/
      COMMON /DRIV/ CAF, CAR, WHBS, WF, WR, U
      SAVE/DRIV/
      COMMON /INOUT/ R, W
      SAVE/INOUT/
      COMMON /TRSSTR/ TTT(4,4,10), TTT1(4,4,10), G(4)
      SAVE/TRSSTR/
C
C  Control Coefficients A, B, C, D defined in section 5.5 of
C   reference (1) and passed from DADS through common block VEHTYP
C   ( A = 1, C=1; B = D = k = 0 defines a conventional front steer
C   vehicle, etc.)
C
      COMMON /VEHTYP/ AAA, BBB, CCC, DDD, RATIO
      SAVE/VEHTYP/
C
C---COMMON Variables------------------------------------------------
C
C   R.....Driver Model Input I/O unit ("DMINPUT.INP")
C   W.....Driver Model Output I/O unit ("DMOUTPUT.OUT")
C
C---DRIV.BLK common block variables---------------------------------
```

```
C
C  CAF...total cornering stiffness of tires on left front susp (lb/rad)
C  CAR...total cornering stiffness of tires on left rear susp (lb/rad)
C  WHBS..wheelbase of vehicle (center-line of front & rear susp (ft)
C  WF....static load on front suspension (lb)
C  WR....static load on rear suspension (lb)
C  U.....initial velocity (ft/sec)
C
C---DRVST1.BLK common block variables-----------------------------------
C
C  GRAV.....gravitational constant
C  TICYCL...driver model sample time (sec)
C  TSS......minimum preview time (sec)
C  DMAX.....upper bound on front wheel angle steer (rad)
C  XP,YP....x-y path coords(SAE) wrt inertial coords (input) (ft)
C  TAUMEM...driver transport time dealy (input parameter) (sec)
C  TFF......driver model preview time (input parameter) (sec)
C  RM.......vehicle mass (slug)
C  A........distance from c.g.  to front suspension center-line (ft)
C  B........distance from c.g.  to rear suspension center-line (ft)
C  RI.......total vehicle yaw inertia (slug-ft)
C  PSIO.....current yaw angle reference value (rad)
C  NTF......number of points in the preview time interval
C  NP.......number of points in the x-y trajectory table
C  TLAST....last time driver model calculated a steer value (sec)
C  DFWLAST..last value of steer calculated by driver model (rad)
C  TILAST...last sample time driver model calculated a steer value (sec)
C  DMEM.....2-dim array (time & steer history) used in delay calculat'n
C  XT,YT....transformation of XP,YP in vehicle body axes (ft)
C
C---TRSSTR.BLK common block variables-----------------------------------
C
C  TTT....transition of matrix at 10 discrete points in preview interval
C  TTT1...integral of trans matrix wrt preview time
C  G......vector of control gain coefficients
C
C---VEHTYP common block variables---------------------------------------
C
C  AAA....Control coefficient A defined in section 5.5 of ref (1)
C  BBB....Control coefficient B defined in section 5.5 of ref (1)
C  CCC....Control coefficient C defined in section 5.5 of ref (1)
C  DDD....Control coefficient D defined in section 5.5 of ref (1)
C  RATIO..rear steer / front steer ratio, k, defined in section 5.5
C
C---Local variables-----------------------------------------------------
C
C  DELT.....time step in local Euler integration (sec)
C  A1.......lat accel coefficient of sideslip veloc in linearized system
C  B1.......   "        yaw rate
C  A2.......yaw accel    "     sideslip vel       "
C  B2.......   "        yaw rate
C  C1.......steer control gain coefficient for lateral accel
C  C2.......steer control gain coefficient for yaw moment
C  ULAST....last value of forward velocity (ft/sec)
C  NBEG.....integer startin counter value
C  NEND1....integer ending counter value
C  NENDV....integer ending counter value
```

```
C  J........integer counter
C  SV.......state vector: y,v,r,yaw,x  (SAE)
C  SV1......integral of state vector
C  SD.......state vector derivative
C
C---Functions and subroutines---------------------------------------------
C
C      None
C
C-------------------------------------------------------------------------
C
C-------------Process Block-----------------------------------------------
C
C
      DELT = 0.01D0
      A1 = -2.  * (CAF + CAR) / RM / U
      B1 = 2.  * (CAR * B - CAF * A) / RM / U - U
      A2 = 2.  * (CAR * B - CAF * A) / RI / U
      B2 = -2.  * (CAR * B * B + CAF * A * A) / RI / U
      C1 = 2.  * (CAF + RATIO * CAR) / RM * AAA + BBB / RM
      C2 = 2.  * (A * CAF - RATIO * B * CAR) / RI * CCC + DDD / RI
      ULAST = U
      G(1) = 0.0D0
      G(2) = C1
      G(3) = C2
      G(4) = 0.0D0
C
      DO 70 J = 1, 4
C
           NBEG = TSS / DELT + 1
           NEND1 = (TFF + .001 - TSS) / NTF / DELT
           NENDV = NEND1
           DO 10 L = 1, 4
                SV(L) = 0.0D0
                SVI(L) = 0.0D0
  10       CONTINUE
      TIME = 0.0D0
C
C  Initialize each state in turn to 1.0 and integrate.
C
C
      SV(J) = 1.0D0
      DO 60 I = 1, NTF
           DO 40 K = NBEG, NENDV
                SD(1) = SV(2) + U * SV(4)
                SD(2) = A1 * SV(2) + B1 * SV(3)
                SD(3) = A2 * SV(2) + B2 * SV(3)
                SD(4) = SV(3)
C
                DO 20 L = 1, 4
                     SV(L) = SV(L) + SD(L) * DELT
  20            CONTINUE
C
                TIME = TIME + DELT
C
                DO 30 L = 1, 4
                     SVI(L) = SVI(L) + SV(L) * DELT
```

```
  30            CONTINUE
C
  40       CONTINUE
C
C  Store "impulse" responses in TTT columns, integral in TTT1.
C  TTT is a NPT-point tabular transition matrix, TTT1 is its integral.
C  (See references 2 & 3.)
C
           DO 50  L = 1, 4
                TTT(L,J,I) = SV(L)
                TTT1(L,J,I) = SVI(L)
  50       CONTINUE
C
           NBEG = NBEG + NEND1
           NENDV = NENDV + NEND1
C
  60       CONTINUE
C
  70  CONTINUE
C
      RETURN
      END
C*********************************************************************


**********************************************************************
**********************************************************************



C
C
      SUBROUTINE Interface_Steering (TIME,TDelta,loc_yd,x_veh,y_veh,
     &                               z_veh,Psi,Theta,Phi,UX,UY,UZ,r,q,
     &                               p,NW,ZLoadR,ZloadL,DFW)
C
      COMMON /DRVST1/ GRAV,TICYCL,TSS,DMAX,XP(100),YP(100),TAUMEM,
     1       TFF,RM,A,B,RI,PSIO,NTF,NP,TLAST,DFWLST,TILAST,
     2       DMEM(1000,2),XT(100),YT(100)
      COMMON /DRIV/ CAF, CAR, WHBS, WF, WR, U
      COMMON/DMTIR/CCAF1,CCAF2,CCAR1,CCAR2,FFZL1,FFZL2,FFZL3,FFZL4,
     +       DMVELC
C
      SAVE/DRIVST1/
      SAVE/DRIV/
      SAVE/DMTIR/
C
      double precision RD(200),TH(200),Y(5),ZLoadR(2),
     &                 ZLoadL(2),SCAN_RANGE(40,40),
     &                 SCAN_PHI(40),SCAN_THETA(40),
     &                 TIME,TDelta,CAF,CAR,
     &                 RM,RI,A,B,
     &                 loc_yd,XV,YV,ZV,Psi,Theta,
     &                 Phi,UX,UY,UZ,r,q,p,DFW,
     &                 RMAX,SMAX_PHI,SMAX_THETA,
     &                 SDELTA_PHI,SDELTA_THETA,THETRAD,
     &                 DELTHRAD,PSIRAD,XSTAR,YSTAR,DFWNOW,
```

```
     &                  U,PI,GRAV, TICYCL, TSS, DMAX,XP,
     &                  YP,TAUMEM,TFF,RM,PSIO,TLAST,DFWLST,
     &                  TILAST,DMEM,XT,YT,WHBS,WF,WR,CCAF1,
     &                  CCAF2,CCAR1,CCAR2,FFZL1,FFZL2,
     &                  FFZL3,FFZL4,DMVELC,DPsi,X_Sen, Y_Sen,
     &                  Z_Sen,Psi_Sen,Theta_Sen,Phi_Sen,
     &                  x_veh,y_veh,z_veh,xstprev
C
      integer NW,kmax,num_theta,k,j,npower,ntf,np
C
      LOGICAL FIRST,flag
C

      DATA FIRST,flag /.TRUE.,.true./
      DATA PI /3.1416/
C
C   Assignments below necessary to avoid using dummy arguments
C    in common block VEHCOORDS .
C
      xv=x_veh
      yv=y_veh
      zv=z_veh
C
      WF = zloadr(1) + zloadl(1)
      WR = zloadr(2) + zloadl(2)
      WHBS = a+b
      CCAF1 = caf/2.0
      CCAF2 = caf/2.0
      CCAR1 = car/2.0
      CCAR2 = car/2.0
      FFZL1 = zloadl(1)
      FFZL2 = zloadr(1)
      FFZL3 = zloadl(2)
      FFZL4 = zloadr(2)
      DMVELC = ux
      Dpsi = r
C
C
      IF(FIRST) THEN
C
C
         CALL Human_Driver (NPOWER,RMAX,SMAX_PHI,
     1                         SMAX_THETA,SDELTA_PHI,
     &                         SDELTA_THETA,
     &                         X_Sen, Y_Sen, Z_Sen,
     &                         Psi_Sen,Theta_Sen,Phi_Sen)
C
         xstprev=xv
C
C
      ENDIF
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCC
CCCCCC  Load the Y vector with vehicle states from DADS.   CCCCCCCCCCCCCCCCCC
CCCCCC                                                     CCCCCCCCCCCCCCCCCC
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      y(1) = YV
      y(2) = loc_yd
      y(3) = r
      y(4) = psi
      y(5) = XV
C
C
         PSIRAD = Psi
         U = UX
C
C
C
         CALL Get_Obstacles (FIRST, TDelta, SMAX_PHI, SMAX_THETA,
     1                       RMAX,SDELTA_PHI, SDELTA_THETA,
     1                       XV, YV, ZV, Psi,Theta, Phi,
     1                       SCAN_RANGE,SCAN_PHI,SCAN_THETA)
C
C
C  Find the minimum range in the set of vertical scans.  Reduce the
C  3D scan to a 2D scan
C
C
       NUM_THETA = INT(SMAX_THETA/SDELTA_THETA)*2 + 1
       KMAX = INT(SMAX_PHI/SDELTA_PHI)*2 + 1
C
       DO 700 K = 1, KMAX
          RD(K) = RMAX
          TH(K) = SCAN_PHI(K)
          DO 800 J = 1, NUM_THETA
C
            IF(SCAN_RANGE(J,K) .LE.  RD(K)) THEN
                   RD(K) = SCAN_RANGE(J,K)
                ENDIF
 800      CONTINUE
 700   CONTINUE
C
C  Interfacing with the original obstacle avoidance subroutine
C
       THETRAD = SMAX_PHI
       DELTHRAD = SDELTA_PHI
       PSIRAD = Psi
C
C
       CALL Avoid_Obstacles (THETRAD, DELTHRAD, NPOWER, XV, YV, ZV,
     1                       PSIRAD, KMAX, RD, TH, XSTAR, YSTAR)
C
C
C  Human steering model
      CALL HUMAN_STEERING(TIME,Tdelta,Y,DFW,DFWNOW,
     &                     XSTAR,YSTAR)
C
      IF(FIRST) THEN
             WRITE(7,705)
  705 FORMAT(" TIME,            XV,            YV,",
     1"            UX           UY             Psi",
```

```
     1"             DPsi,      DFW,          XSTAR,",
     1"       YSTAR")
      ENDIF
      FIRST = .FALSE.
C
C
      return
      END
C



*****************************************************************************
*****************************************************************************



C
C
C Subroutine CALCRS calculates RSTAR, the average value of the range array.
C
      SUBROUTINE CALCRS(IMAX,R,TH,THMAX,DELTH,RSTAR)
C
C -------------------VARIABLE DESCRIPTIONS-----------------------------
C
C -------------------ARGUEMENTS PASSED---------------------------------
C
C IMAX  - Number of sensor increments
C R     - Range array from GENRAY
C TH    - Theta array from GENRAY
C THMAX - Sensor half field of view
C DELTH - Sensor field of view increment
C RSTAR - Previewed range
C
C -----------------------------------------------------------------------
C
      double precision R(*),TH(*),THMAX,DELTH,RSTAR,
     &                 SUM
C
      SUM = 0.0
C
      DO 100 I=1,IMAX
         SUM = SUM + R(I)
 100  CONTINUE
C
      RSTAR = SUM/IMAX
C
      RETURN
      END
C


*****************************************************************************
*****************************************************************************
```

```
C
C
C Subroutine CALCTH calculates THSTAR, the WEIGHTED average of the
C range-theta array.
C
      SUBROUTINE CALCTH(IMAX,R,TH,THMAX,DELTH,NPOWER,THSTAR)
C
C -------------------VARIABLE DESCRIPTIONS-----------------------------
C
C -------------------ARGUEMENTS PASSED---------------------------------
C
C IMAX   - Number of sensor angle increments
C R      - Range array from GENRAY
C TH     - Theta array from GENRAY
C THMAX  - Sensor half field of view
C DELTH  - Sensor field of view increment
C NPOWER - Sensor power
C THSTAR - Weighted average of the range-theta array
C
C ---------------------------------------------------------------------
C
      double precision R(*),TH(*),THMAX,DELTH,THSTAR,
     &                 SUM1,SUM2
C
      SUM1 = 0.0
      SUM2 = 0.0
C
      DO 100 I=1,IMAX
         SUM1 = SUM1 + (R(I)**NPOWER)*TH(I)
         SUM2 = SUM2 + (R(I)**NPOWER)
 100  CONTINUE
C
      THSTAR = SUM1/SUM2
C
      RETURN
      END
C



**********************************************************************
**********************************************************************



C
C
C Subroutine CHECKRTH recalculates RSTAR if RSTAR is greater than the
C range to an obstacle's edge.
C
      SUBROUTINE CHECKRTH(IMAX,R,TH,RSTAR,THSTAR)
C
C -------------------VARIABLE DESCRIPTIONS-----------------------------
C
C -------------------ARGUEMENTS PASSED---------------------------------
C
```

```
C IMAX   - Number of sensor angle increments
C R      - Range array from GENRAY
C TH     - Theta array from GENRAY
C THSTAR - Sensor angle increment corresponding to RSTAR
C RSTAR  - Maximum range within the sensor field of view
C -------------------------------------------------------------------
C
C
      double precision R(200),TH(200),THSTAR,
     &                 A,RANGE,RSTAR
C
      A=RSTAR
C
C From the range-theta profile find the range at THSTAR
C
      DO 100 I=1,IMAX-1
         IF((THSTAR .GE.  TH(I)) .AND.  (THSTAR .LE.  TH(I+1)))THEN
            RANGE=R(I)+((R(I+1)-R(I))/(TH(I+1)-TH(I)))*(THSTAR-TH(I))
         ENDIF
 100  CONTINUE
C
C
C IF RSTAR is greater than the range from the range-theta profile
C reduce RSTAR by 20%
C
C
      IF(A .GE.  RANGE)THEN
         A=0.80*RANGE
      ENDIF
C
      RSTAR=A
C
      RETURN
      END


**********************************************************************
**********************************************************************




C============Algorithm Description=====================================
C
C  Purpose and use:
C   This routine calculates terms associated with the user defined
C   algebraic element.
C
C  Error conditions:     none
C
C  Machine dependencies: none
C
C=====================================================================


      SUBROUTINE FR3512 ( TIME, ENS, FN35, IC, RC, NEL, IEVAL, IMODUL,
     &                    IBLOCK, IFN, ERRCOD, INFOF, UPDATI, SMPSTP,
```

```
     &                    A, IA, TOL )


C============Variable Descriptions=========================================
C
C---Arguments passed------------------------------------------------------
C
C  TIME.......Current simulation time.
C  ENS........Vector of control variable values.
C  FN35.......Vector used if there are algebraically related nodes.
C  IC.........Array of control element integer data.
C  RC.........Array of control element real data.
C  NEL........Number of user algebraic elements in the model.
C  IEVAL......Analysis status flag for controls (=1...6).
C  IMODUL.....Control module currently being processed (=12).
C  IBLOCK.....Number of the user algebraic element currently being
C             processed.  (=1...NEL)
C  IFN........Used if the control system has algebraic loops in the
C             path.
C  ERRCOD.....Error code (zero represents no error).
C  INFOF......File unit for the information file.
C  UPDATI.....  From Rev7 on this flag is no longer used.  User should
C              ignore this flag.
C  SMPSTP.....  From Rev7 on this flag is no longer used.  User should
C              ignore this flag.
C  A..........Array of all real (double precision) data used in the
C             analysis.
C  IA.........Array of all integer data used in the analysis.
C  TOL........If time - sample time is within this tolerance then
C             this is a sample step.
C
      INTEGER  NEL, IC(NEL,*), IEVAL, IMODUL, IBLOCK, IFN, ERRCOD,
     &         INFOF, IA(0:1), GETNOD, INDXAR

      DOUBLE PRECISION  ENS(*), FN35(*), RC(NEL,*), TIME, A(0:1)
      LOGICAL  UPDATI, SMPSTP
      external GETNOD, INDXAR
C
C---COMMON blocks---------------------------------------------------------
      common /vehcoords/ x_veh,y_veh,z_veh,x0,y0,z0
      save/vehcoords/
      save/vehspecs/
C
C---Local variables------------------------------------------------------
C
C    INODn.....Input value of the n'th input node.
C    OUTNOD....Value of the output node after calculations.
C    IVALx.....Integer values passed in from the preprocessor for
C              general use.
C    VALx......Real values similar to IVALx
C
      DOUBLE PRECISION T, INOD1, INOD2, INOD3,INOD4, INOD5,
     &                 INOD6, INOD7, INOD8, OUTNOD, SRATE,
     &                 SMPLOW, SMPHGH, TLSMP, TSMP, VAL1, VAL2,
     &                 VAL3, TOL,xd,yd,zd,zw,zfr(2),zfl(2),roll,
     &                 pitch,yaw,crstfr(2),crstfl(2),mass5,Izz5,
     &                 cgfs,cgrs,pubts,dadsts,cgX,cgY,cgZ,strcom,
```

```
     &                 fstang,vehwid,x,y,z,vmass,xw,yw,ezero,
     &                 eone,etwo,ethree,TDelta,oldtim,ydLoc,
     &                 drvtrq,rfwang,rfww,rrww,x_veh,y_veh,z_veh,
     &                 x0,y0,z0,xprev
C
      INTEGER  IVAL1, IVAL2, IVAL3, numwhl,numrb,numsen,pubrat,i
C
      logical first,flag
C
C---Functions and subroutines-----------------------------------------
C    none
C
C---DATA statements---------------------------------------------------
C
      data vmass,first,oldtim,flag / 0.0,.true.,0.0,.true.  /
C
C=======================================================================

C============Process Block=============================================
      T = TIME
      OUTNOD = 0.0
      INOD1  = ENS(IC(IBLOCK, 1))
      INOD2  = ENS(IC(IBLOCK, 2))
      INOD3  = ENS(IC(IBLOCK, 3))
      INOD4  = ENS(IC(IBLOCK, 4))
      INOD5  = ENS(IC(IBLOCK, 5))
      INOD6  = ENS(IC(IBLOCK, 6))
      INOD7  = ENS(IC(IBLOCK, 7))
      INOD8  = ENS(IC(IBLOCK, 8))
      IVAL1  = IC(IBLOCK, 10)
      IVAL2  = IC(IBLOCK, 11)
      IVAL3  = IC(IBLOCK, 12)
      VAL1   = RC(IBLOCK, 1)
      VAL2   =  RC(IBLOCK, 2)
      VAL3   =  RC(IBLOCK, 3)

      IF ( IEVAL .EQ.  1 ) THEN

C**********************************************************************
C* The user should place the desired calculations below this comment.  *
C* Note that the final output node calculation should be placed in     *
C* OUTNOD.  For instance, if the output node was to be the cubic root  *
C* of the first input node, the calculation would be as follows:       *
C*       OUTNOD = INOD1**(1/3)                                         *
C* Leave the rest of the code alone.                                   *
C*                                                                     *
C*   It is possible that this element maybe digital.  Digital means    *
C*   that the output is delayed by srate.  If the element is digital    *
C*   then IC(IBLOCK,13) is one and the following code is executed.     *
C*   If the element is not digital then this code is ignored.  Note     *
C*   the output node must be calculated before this block of code.     *
C**********************************************************************
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC                                                          CCCC
CCCCCCC   Grab model states (nodes) and prepare to pass them out  CCCC
```

```
CCCCCCC                                                                  CCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
CCCCCCC   DADS outputs units in the following format :         CCCCCCC
CCCCCCC                                                       CCCCCCC
CCCCCCC     Length  -  inch                                    CCCCCCC
CCCCCCC     Mass    -  (lb*sec**2)/inch                       CCCCCCC
CCCCCCC     Force   -  lb (pound)                             CCCCCCC
CCCCCCC                                                       CCCCCCC
CCCCCCC   The following vehicle state units will be changed   CCCCCCC
CCCCCCC    to represent                                       CCCCCCC
CCCCCCC                                                       CCCCCCC
CCCCCCC     Length  in        foot                            CCCCCCC
CCCCCCC     Mass    in        (lb*sec**2)/foot                CCCCCCC
CCCCCCC     Force   remains   lb (pound)                       CCCCCCC
CCCCCCC                                                       CCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      if(first)then
         numwhl = indxar('a',3,31,0,42,ia)
         numrb = indxar('a',3,3,0,1,ia)
         crstfr(1) = a(indxar('a',3,31,1,8,ia))
         crstfr(2) = a(indxar('a',3,31,3,8,ia))
         crstfl(1) = a(indxar('a',3,31,2,8,ia))
         crstfl(2) = a(indxar('a',3,31,4,8,ia))
         mass5 = (a(indxar('a',3,3,1,1,ia)))*12.0
         x0 = (ens(getnod('CHASSIS_X_POS       ')))/12.0
         y0 = (ens(getnod('CHASSIS_Y_POS       ')))/12.0
         z0 = (ens(getnod('CHASSIS_Z_POS       ')))/12.0
         xprev=x0
C     Temporary hardcode for HMMWV Izz5 (slug*ft), mass (slugs), cgfs (ft)and
cgrs (ft)
C       while node value is not setting.
C
C         Izz5 = (a(indxar('a',3,3,1,4,ia)))/12.0
         Izz5 = 7500.0
C         cgfs = (ens(getnod('VEHCG_FRSUSP_DIST   ')))/12.0
         cgfs = 6.8
C         cgrs = (ens(getnod('VEHCG_RRSUSP_DIST   ')))/12.0
         cgrs = 4.0
C         vehwid = (ens(getnod('VEHICLE_WIDTH       ')))/12.0
         do 555 i=1,numrb
            vmass = vmass + (a(indxar('a',3,3,i,1,ia)))*12.0
 555     continue
         first = .false.
      endif
C
      x = (ens(getnod('CHASSIS_X_POS       ')))/12.0
      y = (ens(getnod('CHASSIS_Y_POS       ')))/12.0
      z = (ens(getnod('CHASSIS_Z_POS       ')))/12.0
      x_veh=x
      y_veh=y
      z_veh=z
      xd = (ens(getnod('CHASSIS_X_VEL       ')))/12.0
      yd = (ens(getnod('CHASSIS_Y_VEL       ')))/12.0
```

```
      zd = (ens(getnod('CHASSIS_Z_VEL       ')))/12.0
      xw = ens(getnod('CHASSIS_ROLL_RATE   '))
      yw = ens(getnod('CHASSIS_PITCH_RATE  '))
      zw = ens(getnod('CHASSIS_YAW_RATE    '))
      roll = ens(getnod('CHASSIS_ROLL_ANG     '))
      pitch = ens(getnod('CHASSIS_PITCH_ANG   '))
      yaw = ens(getnod('CHASSIS_YAW_ANG     '))
      zfr(1) = a(indxar('a',3,31,1,42,ia))
      zfr(2) = a(indxar('a',3,31,3,42,ia))
      zfl(1) = a(indxar('a',3,31,2,42,ia))
      zfl(2) = a(indxar('a',3,31,4,42,ia))
      fstang = ens(getnod('FR_STEER_ACT_SENSOR '))
      rfwang = ens(getnod('FR_WHEEL_WRT_CHASSIS'))
      ydLoc  = (ens(getnod('CHASSIS_LOCAL_Y_VEL ')))/12.0
      rfww = ens(getnod('RF_WHEEL_RATE       '))
      rrww = ens(getnod('RR_WHEEL_RATE       '))
C
C
C
      if(t-oldtim.le.0.0)then
         TDelta = 0.001
      else
         TDelta = t-oldtim
      endif
C
C
      if(iblock.eq.1)then
         call VEH_STEER(t,crstfl,crstfr,zfl,zfr,vmass,cgfs,cgrs,Izz5,
     &                  vehwid,y,yd,zw,yw,xw,roll,pitch,yaw,x,
     &                  fstang,rfwang,xd,z,zd,TDelta,ydLoc,numwhl,
     &                  strcom)
         outnod = strcom
       elseif(iblock.eq.2)then
         call speed(rrww,drvtrq)
C         outnod = drvtrq
         outnod = 0.0
      endif
C
      oldtim = t
C
C--Following code only executed if element is digital.

      IF(IC(IBLOCK,13) .EQ.  1)THEN
        SRATE  = RC(IBLOCK,4)
        TLSMP  = RC(IBLOCK,5)

        TSMP   = TLSMP + SRATE
        SMPLOW = TSMP  - TOL
        SMPHGH = TSMP  + TOL

       IF(T .GE.  SMPLOW .AND.  T .LE.  SMPHGH)THEN
         RC(IBLOCK,6) = OUTNOD
       ELSE
         OUTNOD = RC(IBLOCK,6)
       ENDIF
      ENDIF
```

```
C***********************************************************************



C***********************************************************************
C* This should be the end of the user code.  The final output node     *
C* calculation should have been placed in the variable OUTNOD.         *
C***********************************************************************

         ENS(IC(IBLOCK,9)) = OUTNOD

      ELSE IF ( IEVAL .EQ.  2 ) THEN

         WRITE ( INFOF, 100 )
         ERRCOD = 23512
  100    FORMAT(/,'USRALG elements not allowed in algebraic loop.',/ )

      ENDIF

      RETURN
      END


***********************************************************************
***********************************************************************



C***********************************************************************
C
C   ***  Matrix Product Subroutine  ***
C
C  GMPRD:  Computes matrix product
C
C-----------Author and Modification Section----------------------------
C
C  Author:    IBM Scientific Subroutine
C
C  Date written:
C
C  Written on:
C
C  Modifications:  C.  MacAdam
C
C-----------------------------------------------------------------------
C
C-----------Algorithm Description--------------------------------------
C
C  Purpose and use:  R = A B
C
C  Error conditions:
C
C  Machine dependencies:  none
C
C  Called by:  DRIVER
C
```

```
C-----------------------------------------------------------------------
C
      SUBROUTINE GMPRD(A, B, R, N, M, L)
C
C-------------Variable Descriptions-------------------------------------
C
C---Arguments passed:
C
C  A.....N x M matrix
C  B.....M x L matrix
C  R.....N x L resultant matrix = A B product
C  N.....integer row dimension of A
C  M.....integer column dimension of A (or row dimension of B)
C  L.....integer column dimension of B
C
      double precision A(N*M),B(M*L),R(N*L)
C
C---COMMON blocks-----------------------------------------------------
C
C     None
C
C---COMMON Variables--------------------------------------------------
C
C     None
C
C---Local variables---------------------------------------------------
C
C  IR, IK, M, K, L, IR, JI, J, N, IB, IK, etc.....integer counters
C
C---Functions and subroutines-----------------------------------------
C
C     None
C
C-----------------------------------------------------------------------
C
C-------------Process Block---------------------------------------------
C
C
      IR = 0
      IK = -M
      DO 10 K = 1, L
           IK = IK + M
           DO 10 J = 1, N
                IR = IR + 1
                JI = J - N
                IB = IK
                R(IR) = 0.
                DO 10 I = 1, M
                     JI = JI + N
                     IB = IB + 1
  10  R(IR) = R(IR) + A(JI) * B(IB)
      RETURN
      END

**************************************************************************
**************************************************************************
```

```
C************************************************************************
C $Id: main.f,v 1.3.2.1.2.1 1997/12/23 15:55:12 bill Exp $

C MAIN: Entry point for DADS-3D - used to reset array sizes.

C============Author and Modification Section============================
C
C  Author:       Chuck Mead
C
C  Date written: January 29, 1986
C
C  Written on:   MicroVAX II
C
C  Modifications:
C    1) 1/28/88   Changed the /AIA/ common block from named common to
C                 blank common to avoid any potential problems resul-
C                 ting from different sizes of the same common block
C                 in different routines.  (Chuck Mead)
C    2) 4/5/88    Increased A and IA to 700000 and 150000.  (Dick Kading)
C $Log: main.f,v $
C Revision 1.3.2.1.2.1  1997/12/23 15:55:12  bill
C Added a comment for NSAVE13.
C
c Revision 1.3.2.1  1996/07/26  16:33:50  alan
c Replaced the explicit definition of the A and IA arrays with the common
block
c 'dadsaia.blk'.  This common block is already being used by several routines
c (particularly in the MatLab and MatrixX interface routines), and Chuck says
c it's OK.
c
c Revision 1.3  1995/10/23  17:39:17  bill
c Define number of data points stored for continuous delay element here.
c This allows user to increase the size.
c
c Revision 1.2  1995/03/31  14:54:41  chuck
c The big 7.6 commit.
c
c Revision 1.1.1.1.10.1  1995/03/23  18:09:28  chuck
c Changed A and IA arrays from blank common to the named common blocks
c used with the DADS/Plant interfaces.
c
C  Copyright (c) CADSI 1988-1995
C
C========================================================================

C============Algorithm Description=====================================
C
C  Purpose and use:
C    This subroutine is the entry point for the DADS-3D analysis pro-
C    gram.  The sizes of the two main arrays, A and IA, are defined
C    here.  To change the size of either the integer array IA or the
C    double precision array A, change the value of the associated
C    parameter, i.e., PIASIZ for IA or PASIZ for A.  Then, recompile
C    this routine, replace it in the MOD3D object library, and relink
```

```
C    DADS-3D.
C
C  Error conditions:     none
C
C  Machine dependencies: none
C
C=======================================================================


      SUBROUTINE MAINA


C============Variable Descriptions=====================================
C
C---Arguments passed-------------------------------------------------
C    none
C
C---COMMON blocks----------------------------------------------------
C

C
C DADSAIA:  Common blocks for A and IA arrays
C
C---Main real & integer data arrays
C
C    A........Vector for all real data in the system.
C    IA.......Vector for all integer data in the system.
C    PASIZ....Parameter defining the upper bound of the A array.
C    PIASIZ...Parameter defining the upper bound of the IA array.
C    PLASIZ...Parameter used to size A array for DADS/plant runs.
C    PLIASIZ...Parameter used to size IA  array for DADS/plant runs.
C
C Copyright (c) CADSI 1996
C=======================================================================




      INTEGER  PASIZ, PIASIZ
      INTEGER  PLASIZ, PLIASIZ
      PARAMETER  ( PASIZ=4000000, PIASIZ=800000 )

      INTEGER           IA(0:PIASIZ)
      DOUBLE PRECISION  A(0:PASIZ)

      COMMON  /DADSIA/  IA
      COMMON  /DADSA /  A
      COMMON  /PLDADSA / PLASIZ, PLIASIZ

      SAVE /DADSIA/, /DADSA/, /PLDADSA/
C
      INTEGER NSAVE13
      COMMON/C8CONDLY/NSAVE13
      SAVE/C8CONDLY/

C
```

```
C---Local variables-----------------------------------------------------
C
C    ASIZE....Local variable for upper bound array declaration for the
C             array A.
C    IASIZE...Local variable for upper bound array declaration for the
C             array IA.
C     NSAVE13..Maximum number of points saved in the control analaog delay
C              element.  The user may increase this if needed.
C
      INTEGER  ASIZE, IASIZE
C
C---Functions and subroutines-------------------------------------------
C
      EXTERNAL  MAINB
C
C=================================================================

C============Process Block=======================================

C---Set the value of the two array sizes equal to the declared
C   parameters.

      ASIZE  = PASIZ
      IASIZE = PIASIZ

C--Number of data points in continuos delay element.

       NSAVE13 = 5000

C---Continue the analysis.

      CALL MAINB ( A, IA, ASIZE, IASIZE )


      RETURN
      END
C


**************************************************************************
**************************************************************************



      subroutine SPEED(ww,drvtrq )
C
      double precision gain,ww,spderr,drvtrq
C
      gain = 10000.0D0
      spderr = 60.0 - ww
C
      drvtrq = spderr*gain
C
      if(drvtrq.le.-20000.0)then
         drvtrq = -20000.0
      elseif(drvtrq.ge.40000.0)then
         drvtrq = 40000.0
```

```
      endif
C
      return
      end
C
*************************************************************************
*************************************************************************
C
C Subroutine TRANXY transforms polar RSTAR and THSTAR to inertial XSTAR
C and YSTAR
C
      SUBROUTINE TRANXY (X,Y,PSI,RSTAR,THSTAR,XSTAR,YSTAR)
C
C -------------------VARIABLE DESCRIPTIONS-----------------------------
C
C -------------------ARGUEMENTS PASSED---------------------------------
C
C X      - Vehicle's forward displacement, (ft)
C Y      - Vehicle's lateral displacement, (ft)
C PSI    - Vehicle's yaw angle, (rad)
C RSTAR  - Average of the range array or previewed range
C THSTAR - Weighted average of the theat array or previewed angle
C XSTAR  - Previewed forward location (ft)
C YSTAR  - Previewed lateral location (ft)
C
C -------------------------------------------------------------------
C
      DOUBLE PRECISION X,Y,PSI,RSTAR,THSTAR,XSTAR,YSTAR
C
      XSTAR=X+RSTAR*COS(PSI+THSTAR)
      YSTAR=Y+RSTAR*SIN(PSI+THSTAR)
C
C
      RETURN
      END
C


*************************************************************************
*************************************************************************
C
      subroutine VEH_STEER(t,crstfl,crstfr,zfl,zfr,vmass,cgfs,cgrs,
     &                     Izz5,vehwid,y,yd,zw,yw,xw,roll,pitch,
     &                     yaw,x,fstang,rfwang,xd,z,zd,TDelta,
     &                     ydLoc,numwhl,strcom)
C
      COMMON /DRVST1/ GRAV,TICYCL,TSS,DMAX,XP(100),YP(100),TAUMEM,
     1       TFF,RM,A,B,RI,PSIO,NTF,NP,TLAST,DFWLST,TILAST,
     2       DMEM(1000,2), XT(100), YT(100)
      COMMON /DRIV/ CAF, CAR, WHBS, WF, WR, U
C
      SAVE/DRIVST1/
      SAVE/DRIV/
C
      double precision t,TDelta,strcom,newang,DFW,oldang,stgain,
     &                  fstang,xd,xdglim,crstfl(2),crstfr(2),zfl(2),
     &                  zfr(2),vmass,cgfs,cgrs,Izz5,y,yd,zw,yaw,x,
```

```
     &                   vehwgt,vehwid,strlim,pi,z,roll,pitch,yw,xw,
     &                   ydLoc,caf,car,grav,ticycl,tss,dmax,xp,yp,
     &                   taumem,tff,rm,a,b,ri,psio,tlast,dfwlst,tilast,
     &                   dmem,xt,yt,whbs,wf,wr,u,zd,rfwang
      integer numwhl,numaxl,ntf,np
C
      data stgain,xdglim,DFW,strlim,pi /50000.0,20.0,0.0,34.0,3.14159/
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                         C
C  Strcom (steer command) is a torque in inch*lbs, stgain (steer gain)    C
C  is in inch*lbs/radian, xd ( x direction velocity) is in               C
C inches/sec, xdglim (x direction velocity gain limit) is in              C
C inches/sec, newang (updated average steer angle) is in radians,         C
C and oldang (previous average steer angle) is in radians.                C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      oldang = fstang
      vehwgt = vmass*32.2
      numaxl = numwhl/2
      caf = crstfl(1)+crstfr(1)
      car = crstfl(2)+crstfr(2)
      rm = vmass
      ri = Izz5
      a = cgfs
      b = cgrs
C
     if(xd.ge.3.0)then
        call Interface_Steering (t,TDelta,ydLoc,x,y,z,yaw,pitch,
     &                         roll,xd,yd,zd,zw,yw,xw,numaxl,
     &                         zfr,zfl,DFW)
      endif
C
C
      newang = DFW
C
C
      strcom = -1.0*(stgain*(newang - rfwang))
C
      return
      end
```

****
************************* END OF FORTRAN CODE ***************************
****

INTENTIONALLY LEFT BLANK.

NO. OF
COPIES ORGANIZATION

* ADMINISTRATOR
DEFENSE TECHNICAL INFO CTR
ATTN DTIC OCA
8725 JOHN J KINGMAN RD STE 0944
FT BELVOIR VA 22060-6218
*pdf file only

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL CI IS R REC MGMT
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL CI OK TECH LIB
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL D D SMITH
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL SE RM E BURKE
G GOLDMAN
AMSRD ARL SE DC A GOLDBERG
2800 POWDER MILL RD
ADELPHI MD 20783-1197

1 DOD JOINT CHIEFS OF STAFF
ATTN J39 CAPABILITIES DIV
CPT J M BROWNELL
THE PENTAGON RM 2C865
WASHINGTON DC 20301

1 DIRECTOR CIA
ATTN D MOORE
WASHINGTON DC 20505-0001

3 PM ABRAMS TANK SYSTEM
ATTN SFAE GCS AB COL KOTCHMAN
P LEITHEISER H PETERSON
WARREN MI 48397-5000

1 PM M1A2
ATTN SFAE GCS AB LTC R LOVETT
WARREN MI 48397-5000

NO. OF
COPIES ORGANIZATION

1 PM M1A1
ATTN SFAE GCS AB LTC L C MILLER JR
WARREN MI 48397-5000

1 PEO-GCS
BRADLEY FIGHTING VEHICLES
ATTN M KING
WARREN MI 48397-5000

1 PM BFVS
ATTN ATZB BV COL C BETEK
FORT BENNING GA 31905

1 PM M2/M3 BFVS
ATTN SFAE GCS BV LTC J MCGUINESS
WARREN MI 48397-5000

3 PM BCT
ATTN SFAE GCS BCT COL R D OGG JR
J GERLACH T DEAN
WARREN MI 48397-5000

1 PM IAV
ATTN SFAE GCS BCT LTC J PARKER
WARREN MI 48397-5000

1 PM NIGHT VISION/RSTA
ATTN SFAE IEW&S NV COL BOWMAN
10221 BURBECK RD
FT BELVOIR VA 22060-5806

1 NIGHT VISION & ELEC SENSORS DIR
ATTN DR A F MILTON
10221 BURBECK RD SUITE 430
FT BELVOIR VA 22060-5806

2 CDR US ARMY TRADOC
ATTN ATINZA R REUSS
ATIN I C GREEN
BLDG 133
FT MONROE VA 23651

1 OFC OF THE SECY OF DEFENSE
CTR FOR COUNTERMEASURES
ATTN M A SCHUCK
WSMR NM 88002-5519

1 US SOCOM
ATTN SOIO JA F J GOODE
7701 TAMPA POINT BLVD BLDG 501
MCDILL AFB FL 33621-5323

NO. OF COPIES ORGANIZATION

1 CDR US ARMY ARMOR CTR & FT KNOX
TSM/ABRAMS
ATTN COL D SZYDLOSKI
FT KNOX KY 40121

1 CDR US AMBL
ATTN COL J JUGHES
FT KNOX KY 40121

1 DIR OF COMBAT DEVELOPMENT
ATTN ATZK FD W MEINSHAUSEN
BLDG 1002 ROOM 326
1ST CAVALRY DIV RD
FT KNOX KY 40121-9142

1 COMMANDING OFFICER
MARINE CORPS INTEL ACTIVITY
ATTN COL WILLIAM BARTH
3300 RUSSELL ROAD SUITE 250
QUANTICO VA 22134-5011

4 CDR US TACOM-ARDEC
ATTN AMSTA AR TD M DEVINE
M FISETTE
AMSTA AR FSA M J FENECK
AMSTA AR FSA P D PASCUA
PICATINNY ARSENAL NJ 07806-5000

4 CDR US TACOM-ARDEC
ATTN AMSTA AR FSA S R KOPMANN
H KERWIEN K JONES
A FRANCHINO
PICATINNY ARSENAL NJ 07806-5000

4 CDR US TACOM-ARDEC
ATTN AMSTA AR FSA T A LAGASCA
AMSTA AR FSP D LADD
M CILLI M BORTAK
PICATINNY ARSENAL NJ 07806-5000

3 CDR US TACOM-ARDEC
ATTN AMSTA AR FSP G A PEZZANO
R SHORR
AMSTA AR FSP I R COLLETT
PICATINNY ARSENAL NJ 07806-5000

7 CDR US TACOM-ARDEC
ATTN AMSTA AR CCH A M PALTHINGAL
A VELLA E LOGSDON
R CARR M MICOLICH
M YOUNG A MOLINA
PICATINNY ARSENAL NJ 07806-5000

NO. OF COPIES ORGANIZATION

3 CDR US TACOM-ARDEC
ATTN AMSTA AR QAC R SCHUBERT
AMSTA AR WE C R FONG S TANG
PICATINNY ARSENAL NJ 07806-5000

1 SAIC
ATTN K A JAMISON
PO BOX 4216
FT WALTON BEACH FL 32549

4 PEO-GCS
ATTN SFAE GCS C GAGNON
SFAE GCS W A PUZZUOLI
SFAE GCS BV J PHILLIPS
SFAE GCS LAV COL T LYTLE
WARREN MI 48397-5000

4 PEO-GCS
ATTN SFAE GCS AB SW DR PATTISON
SFAE GCS AB LF LTC PAULSON
SFAE GCS LAV M T KLER
SFAE GCS LAV FCS MR ASOKLIS
WARREN MI 48397-5000

3 CDR US ARMY TACOM
ATTN AMSTA TR DR R MCCLELLAND
MR BAGWELL
AMSTA TA J CHAPIN
WARREN MI 48397-5000

12 CDR US ARMY TACOM
ATTN AMSTA TR R DR J PARKS C ACIR
S SCHEHR D THOMAS J SOLTESZ
S CAITO K LIM J REVELLO
B BEAUDOIN B RATHGEB
M CHAIT S BARSHAW
WARREN MI 48397-5000

8 CDR US ARMY TACOM
ATTN AMSTA CM XSF R DRITLEIN
MR HENDERSON MR HUTCHINSON
MR SCHWARZ S PATHAK
R HALLE J ARKAS G SIMON
WARREN MI 48397-5000

5 PEO PM MORTAR SYSTEMS
ATTN SFAE AMO CAS IFM L BICKLEY
M SERBAN K SLIVOVSKY
SFAE GCS TMA R KOWALSKI
SFAE GCS TMA PA E KOPACZ
PICATINNY ARSENAL NJ 07860-5000

NO. OF COPIES ORGANIZATION

3 MIT LINCOLN LABORATORY
ATTN J HERD G TITI D ENGREN
244 WOOD STREET
LEXINGTON MA 02420-9108

2 THE UNIV OF TEXAS AT AUSTIN
INST FOR ADVANCED TECH
ATTN I MCNAB S BLESS
PO BOX 20797
AUSTIN TX 78720-2797

1 INNOVATIVE SURVIVABILITY TECH
ATTN J STEEN
PO BOX 1989
GOLETA CA 93116

1 SUNY BUFFALO
ELECTRICAL ENGINEERING DEPT
ATTN J SARJEANT
PO BOX 601900
BUFFALO NY 14260-1900

1 GENERAL DYNAMICS LAND SYSTEMS
ATTN D GERSDORFF
PO BOX 2074
WARREN MI 49090-2074

1 CDR US ARMY CECOM
ATTN W DEVILBISS
BLDG 600
FT MONMOUTH NJ 07703-5206

1 MARCORSYSCOM/CBG
ATTN CPT J DOUGLAS
QUANTICO VA 22134-5010

2 CDR USAIC
ATTN ATZB CDF MAJ J LANE
D HANCOCK
FT BENNING GA 31905

2 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL SL EA R CUNDIFF
AMSRD ARL SL EM J THOMPSON
WSMR NM 88001-5513

4 UNITED DEFENSE ADV DEV CTR
ATTN K GROVES J FAUL T WINANT
V HORVATICH
328 BROKAW ROAD
SANTA CLARA CA 95050

NO. OF COPIES ORGANIZATION

2 NORTHROP GRUMMAN CORP
ATTN A SHREKENHAMER D EWART
1100 W HOLLYVALE STREET
AAUSA CA 91702

1 CDR US ARMY AMCOM
ATTN AMSAM RD ST WF D LOVELACE
REDSTONE ARSENAL AL 35898-5247

1 OFC OF THE SECY OF DEFENSE
ATTN ODDRE (R&T) G SINGLEY
THE PENTAGON
WASHINGTON DC 20301-3080

1 US MILITARY ACADEMY
MATH SCIENCES CTR OF EXCELLENCE
DEPT OF MATHEMATICAL SCIENCES
ATTN MDN A MAJ HUBER
THAYER HALL
WEST POINT NY 10996-1786

1 DIR US ARMY WATERWAYS EXPER STN
ATTN R AHLVIN
3909 HALLS FERRY ROAD
VICKSBURG MS 39180-6199

1 NATL INST STAN AND TECH
ATTN K MURPHY
100 BUREAU DRIVE
GAITHERSBURG MD 20899

1 CDR US ARMY MMBL
ATTN MAJ J BURNS
BLDG 2021
BLACKHORSE REGIMENT DRIVE
FT KNOX KY 40121

2 DIRECTOR
NASA JET PROPULSION LAB
ATTN L MATHIES K OWENS
4800 OAK GROVE DRIVE
PASADENA CA 91109

1 DIRECTOR
AMCOM MRDEC
ATTN AMSMI RD W C MCCORKLE
REDSTONE ARSENAL AL 35898-5240

1 COMMANDER
CECOM
SP & TERRESTRIAL COM DIV
ATTN AMSEL RD ST MC M H SOICHER
FT MONMOUTH NJ 07703-5203

NO. OF
COPIES ORGANIZATION

1 COMMANDER
US ARMY INFO SYS ENGRG CMD
ATTN ASQB OTD F JENIA
FT HUACHUCA AZ 85613-5300

1 COMMANDER
US ARMY NATICK RDEC
ACTING TECHNICAL DIR
ATTN SSCNC T P BRANDLER
NATICK MA 01760-5002

1 COMMANDER
ARMY RESEARCH OFC
4300 S MIAMI BLVD
RSCH TRIANGLE PARK NC 27709

1 COMMANDER
US ARMY STRICOM
ATTN J STAHL
12350 RSCH PARKWAY
ORLANDO FL 32826-3726

1 COMMANDER
US ARMY TRADOC
BATTLE LAB INTEGRATION 7 TECH DIR
ATTN ATCD B J A KLEVECZ
FT MONROE VA 23651-5850

1 COMMANDER
ATTN CODE B07 J PENNELLA
17320 DAHLGREN ROAD
BLDG 1470 RM 1101
DAHLGREN VA 22448-5100

1 DARPA
3701 N FAIRFAX DRIVE
ARLINGTON VA 22203-1714

1 COMMANDER
US ARMY AVIATION & MISSILE CMD
ATTN AMSAM-RD-SS-EG A KISSELL
BLDG 5400
REDSTONE ARSENAL AL 35898

1 OFC OF THE PROJECT MGR
MANEUVER AMMUNITION SYSTEMS
ATTN S BARRIERES
BLDG 354
PICATINNY ARSENAL NJ 07806-5000

1 COMMANDER
US ARMY TRADOC ANALYSIS CTR
ATTN ATRC-WBA J GALLOWAY
WSMR NM 88002-5502

NO. OF
COPIES ORGANIZATION

1 FASTTRACK TECH INC
ATTN J K GARRETT
540 CEDAR DRIVE
RADCLIFF KY 40160

1 DIR USARMY TACOM
6501 E ELEVEN MILE RD
WARREN MI 48397-5000

ABERDEEN PROVING GROUND

2 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL CI OK (TECH LIB)
BLDG 305 APG AA

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL HR SC
BLDG 459

1 CDR US ARMY EDGEWOOD RDEC
ATTN SCBRD TD J VERVIER
APG EA

2 CDR US ARMY TECOM
ATTN AMSTE CD B SIMMONS
AMSTE CD M R COZBY
RYAN BLDG

4 DIR US AMSAA
ATTN AMXSY D M MCCARTHY
P TOPPER
AMXSY CA G DRAKE S FRANKLIN
BLDG 367

7 CDR US ATC
ATTN CSTE AEC COL ELLIS
CSTE AEC TD J FASIG
CSTE AEC TE H CUNNINGHAM
CSTE AEC RM C A MOORE
CSTE AEC TE F P OXENBERG
A SCRAMLIN
CSTE AEC CCE W P CRISE
BLDG 400

1 PM ODS
ATTN SFAE CBD COL B WELCH
BLDG 4475
APG EA

NO. OF
COPIES ORGANIZATION

5 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM J SMITH
E SCHMIDT B RINGER
T ROSENBERGER
B BURNS
BLDG 4600

3 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM
C SHOEMAKER
J BORNSTEIN
AMSRD ARL WM BF J WALL
BLDG 1121

2 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM B A HORST
W CIEPIELLA
BLDG 4600

3 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM BA D LYONS
AMSRD ARL WM BC P PLOSTINS
AMSRD ARL WM BD B FORCH
BLDG 4600

2 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM MB L BURTON
BLDG 4600

7 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM BF T HAUG
P FAZIO R PEARSON
M FIELDS G HAAS
W OBERLE J WALD
BLDG 390

7 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM TE A NIILER
G THOMSON T KOTTKE
M MCNEIR P BERNING
J POWELL C HUMMER
BLDG 120

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL WM TC R COATES
BLDG 309

NO. OF
COPIES ORGANIZATION

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL SL BG M ENDERLEIN
BLDG 247

1 DIRECTOR
US ARMY RSCH LABORATORY
ATTN AMSRD ARL SL EM C GARRETT
BLDG 390A